ВИЛЬЯМС
OSBORNE

Полный справочник

по

Оригинал был взять с medigo.ru
PDF2GIFs by Kvait
GIFs2DJVU by karpinskiy1
Для http://netz.ru/

Родерик Смит

FreeBSD: The Complete Reference

Roderick W Smith

McGraw-Hill/Osborne

New York Chicago San Francisco
Lisbon London Madrid Mexico City
Milan New Delhi San Juan
Seoul Singapore Sydney Toronto

Полный справочник по FreeBSD

Родерик Смит

Издательский дом "Вильямс"
Москва • Санкт-Петербург • Киев
2004

ББК 32.973.26 018 2 75
С50
УДК 681.3 07

Издательский дом "Вильямс"
Зав. редакцией *С.Н. Тригуб*

Перевод с английского и редакция *В Р Гинзбурга*

По общим вопросам обращайтесь в Издательский дом "Вильямс" по адресу
info@williamspublishing.com, http://www williamspublishing com

Смит Родерик
С50 Полный справочник по FreeBSD Пер с англ — М Издательский дом "Вильямс", 2004 — 672 с. : ил — Парал тит англ

ISBN 5 8459-0576-1 (рус.)

Эта книга может служить подробным руководством по инсталляции конфигурированию и обслуживанию популярной операционной системы FreeBSD. Читатели узнают, как интегрировать эту мощную операционную систему с существующим аппаратным и программным обеспечением, как управлять ею и как использовать имеющиеся в ней графические и офисные приложения.

В книге делается упор на администрирование FreeBSD, а не на обычное использование системы, поэтому руководство ориентировано в основном на администраторов FreeBSD-систем, но будет полезно и пользователям устанавливающим FreeBSD для личных целей

ББК 32 973 26 018 2 75

Все названия программных продуктов являются зарегистрированными торговыми марками соответствующих фирм

Никакая часть настоящего издания ни в каких целях не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами, будь то электронные или механические, включая фотокопирование и запись на магнитный носитель если на это нет письменного разрешения издательства Osborne Publishing.

Authorized translation from the English language edition published by Osborne Publishing Copyright © 2003 by McGraw-Hill Companies

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Russian language edition published by Williams Publishing House according to the Agreement with R&I Enterprises International Copyright © 2004

ISBN 5-8459-0576-1 (рус.)
ISBN 0-07-222409-6 (англ.)

© Издательский дом "Вильямс 2004
© McGraw-Hill Companies, 2003

Оглавление

Введение 24

Часть I. Инсталляция FreeBSD 31
Глава 1. Системные требования 32
Глава 2. Инсталляция системы 55
Глава 3. Обзор системы 85
Глава 4. Сосуществование с другими операционными системами 108

Часть II. Основы системного администрирования 133
Глава 5. Средства системного администрирования 134
Глава 6. Запуск системы и управляющие процессы 156
Глава 7. Управление разделами 174
Глава 8. Управление файлами 192
Глава 9. Конфигурирование принтера 216
Глава 10. Управление учетными записями 236
Глава 11. Инсталляция программного обеспечения 255
Глава 12. Конфигурирование ядра 272
Глава 13. X Window System 281

Часть III. Сетевое конфигурирование 301
Глава 14. Основы сетевого конфигурирования 302
Глава 15. Коммутируемые сети 318
Глава 16. Сетевые клиенты и серверы 331
Глава 17. Брандмауэры 343

Часть IV. Серверы 361
Глава 18. Файловые серверы 362
Глава 19. Почтовые серверы 376
Глава 20. Web серверы 392
Глава 21. Серверы дистанционной регистрации 405
Глава 22. Вспомогательные серверы 429

Часть V. Пользовательское программное обеспечение 449
Глава 23. Настольные графические оболочки 450
Глава 24. Сетевые утилиты 468
Глава 25. Офисные средства 491
Глава 26. Графические средства 510
Глава 27. Мультимедийные и игровые приложения 528

Часть VI. Обслуживание системы 547
Глава 28. Автоматизированные и неавтоматизированные процедуры обслуживания 548
Глава 29. Системная безопасность 568
Глава 30. Компиляция программ 594
Глава 31. Написание сценариев 605
Глава 32. Поиск и устранение неполадок 626
Словарь терминов 646
Предметный указатель 664

Содержание

Введение 24

Часть I. Инсталляция FreeBSD 31

Глава 1. Системные требования 32
Для чего нужна FreeBSD 33
FreeBSD в роли настольной ОС 33
FreeBSD в роли серверной ОС 34
FreeBSD и неродственные операционные системы 36
FreeBSD и другие разновидности UNIX 37
Программная среда 40
История разработки FreeBSD 40
Взаимодействие с другими компьютерами по сети 42
Сосуществование с другими операционными системами на одном компьютере 43
Аппаратная среда 45
Типы и производительность процессоров 45
Требования к оперативной памяти 46
Жесткие диски 47
Диски SCSI и EIDE 47
Общие требования к объему дискового пространства 48
Графическое оборудование 50
Сетевое оборудование 51
Принтеры 52
Вспомогательные устройства 53
Резюме 54

Глава 2. Инсталляция системы 55
Получение FreeBSD 56
Лицензирование 56
Источники 57
Подготовка инсталляционного носителя 58
Загрузочные диски 58
Источник инсталляции 59
Подготовка компьютера к запуску FreeBSD 59
Проверка аппаратной совместимости 60
Очистка дискового пространства 62
Программа FIPS 63
Программа PartitionMagic 64
Начало инсталляции 65
Загрузка инсталлятора 65
Разбивка на разделы 66
Создание первичного раздела FreeBSD 67
Задание параметров загрузчика 68
Создание подразделов 68
Выбор инсталлируемого программного обеспечения 70
Среда инсталляции 71
Непосредственная установка системы 72
Начальное конфигурирование системы 72

Настройка параметров сети 72
Установки системной консоли 74
Часовой пояс 75
Совместимость с исполняемыми файлами Linux 75
Настройка параметров мыши 75
Конфигурирование пакета XFree86 76
Конфигурирование рабочего стола X-среды 79
Конфигурирование программных пакетов 80
Настройка учетных записей 80
Значение учетных записей 80
Создание группы 82
Создание учетной записи 83
Завершение инсталляции 84
Резюме 84
Глава 3. Обзор системы 85
Начальная загрузка 86
Выбор FreeBSD 86
Интерпретация сообщений ядра 87
Сценарии запуска системы 88
Регистрация в системе 89
Регистрация в текстовом и графическом режимах 89
Обычные и суперпользовательские учетные записи 91
Команды текстового режима 92
Интерпретаторы команд 92
Запуск программ в текстовом режиме 94
Команды манипулирования файлами 95
cd: изменение текущего каталога 95
ls: списки файлов 95
cp: копирование файлов 97
mv: перемещение файлов 97
rm: удаление файлов 97
mkdir: создание каталога 98
rmdir: удаление каталога 98
Краткий обзор графической среды 98
Переход в графический режим 98
Управление файлами 100
Запуск программ в графической среде 102
Настройка графической среды 102
Выход из системы и останов 104
Выход из системы в текстовом и графическом режимах 104
Останов системы в текстовом и графическом режимах 105
Когда нужно останавливать систему 106
Резюме 107
Глава 4. Сосуществование с другими операционными системами 108
Настройка процесса начальной загрузки 109
Системные загрузчики 109
Использование загрузчика FreeBSD 110
Задание параметров ядра 112
Использование стороннего загрузчика 113
Управление разделами FreeBSD и x86 114
Типы разделов 114

Защита разделов FreeBSD от повреждения другими операционными системами 115
Доступ к внешним разделам из FreeBSD 117
Файловые системы 118
Два значения термина "файловая система" 118
Ключевые особенности файловой системы FreeBSD 118
Другие важные файловые системы 119
Разновидности файловой системы FreeBSD 119
FAT 120
NTFS 121
Ext2fs 121
Журнальные файловые системы 121
Файловые системы компакт-дисков 122
Доступ к внешним файловым системам из FreeBSD 122
Монтирование внешних файловых систем 123
Ограничения на использование внешних файловых систем 124
Доступ к файловым системам FreeBSD из других операционных систем 125
Поддержка со стороны других операционных систем 125
Особенности внешнего доступа 126
Запуск сторонних программ во FreeBSD 127
Запуск программ Linux 127
Подготовка к запуску 127
Переконфигурирование и запуск 128
Использование пакета WINE для запуска приложений Windows 129
Инсталляция и конфигурирование пакета WINE 129
Запуск приложений Windows 131
Использование эмулятора 132
Резюме 132

Часть II. Основы системного администрирования 133

Глава 5. Средства системного администрирования 134
Каталог /etc и его содержимое 135
Сценарии запуска системы 135
Серверные конфигурационные файлы 136
Вспомогательные конфигурационные файлы 138
Текстовые редакторы 138
Редакторы текстового режима 139
Редакторы графического режима 140
Пример: редактирование файла в vi 140
Режимы работы 140
Редактирование файла 141
Сохранение изменений 143
Пример: редактирование файла в gEdit 143
Редактирование файла 144
Дополнительные возможности 145
Важные административные команды 146
Команды для получения информации о системе 146
dmesg 146
df 147
man 148
uptime 148
ps 149
top 150

Команды манипулирования системой 150
nice и renice 150
kill и killall 151
Переадресация ввода вывода 153
Каналы 154
Графические административные утилиты 154
Резюме 155
Глава 6. Запуск системы и управляющие процессы 156
Системное ядро 158
Загрузка ядра 158
Влияние ядра на последующие этапы загрузки системы 158
Роль ядра в управлении работающей системой 158
Настройка сценариев запуска системы 160
Редактирование системных сценариев 160
Создание новых локальных сценариев 161
Особенности регистрации в системе 162
Методы регистрации 162
Консольная регистрация в текстовом режиме 162
Консольная и дистанционная регистрация в графическом режиме 162
Дистанционная регистрация в текстовом режиме 163
Специальные серверы 164
Аутентификация пользователей с помощью модулей PAM 164
Настройка стандартных параметров регистрации 166
Пользовательские конфигурационные файлы 167
Сценарии регистрации в текстовом режиме 167
Сценарии регистрации в графическом режиме 168
Запуск программ в запланированное время: демон cron 169
Команда at 169
Функции демона cron 170
Планирование системных заданий 171
Планирование пользовательских заданий 172
Резюме 173
Глава 7. Управление разделами 174
Разделы: хранилища файлов данных 175
Зачем разбивать диск на разделы 175
Создание раздела 176
Использование утилит сторонних производителей 176
Использование программы sysinstall 176
Использование утилит командной строки 178
Монтирование разделов 182
Команда mount 183
Файл /etc/fstab 184
Демонтирование разделов 186
Традиционные схемы разбивки на разделы 186
Какие каталоги могут быть разделами? 186
Распространенные схемы разбивки на разделы 187
Монтирование внешних разделов 187
FHS 188
Роль FHS 188
Обзор каталогов FHS 189
Уровень совместимости с FHS 191
Резюме 191

Глава 8 Управление файлами 192
Права доступа и принадлежность файлов 193
Учетные записи и владельцы файлов 193
Контроль доступа к файлам 194
Особенности доступа к каталогам 197
Разработка общесистемных правил доступа 197
Команды манипулирования файлами 198
Изменение владельца файла 198
Изменение прав доступа 199
Копирование и перемещение файлов 201
Команда cp 201
Команда mv 202
Команда cat 202
Команда dd 203
Удаление файлов 204
Ссылки на файлы 205
Поиск файлов 206
Команда find 207
Команды grep и egrep 208
Резервное копирование 210
Резервные носители 210
Утилиты резервного копирования 211
Утилита tar 211
Восстановление данных 214
Резюме 215
Глава 9. Конфигурирование принтера 216
Модель печати во FreeBSD 217
Очередь на печать 217
Постановка заданий в очередь 218
Демон построчной печати 219
Язык PostScript 219
Создание очереди на печать 220
Файлы устройств печати 220
Тестирование файлов устройств 221
Оптимизация параллельного порта 222
Замечания по поводу сетевых принтеров 222
Создание каталога спулинга 222
Формат файла /etc/printcap 223
Использование простейшего фильтра 225
Конфигурирование программы Ghostscript 226
Использование программы Ghostscript в роли транслятора 227
Настройка сложного фильтра 228
Управление принтером 229
Выдача заданий на печать 229
Команда lpr 230
Печать из приложений 230
Контроль заданий 231
Удаление заданий 232
Переупорядочивание заданий 233
Изменение статуса очереди 233
Контроль доступа к сетевому принтеру 234
Резюме 235

Глава 10. Управление учетными записями 236
Роль учетных записей во FreeBSD 237
Упорядочение работы пользователей 237
Безопасность системы 238
Группы пользователей 238
Имена и идентификаторы пользователей и групп 239
Структуры данных, связанные с учетными записями 240
Создание учетных записей 240
Правила именования пользователей 241
Использование программы sysinstall 241
Использование утилиты adduser 243
Модификация учетных записей 244
Скрытые пароли 245
Команда passwd 245
Команда chpass 246
Файл /etc/master.passwd 247
Изменение идентификаторов пользователей 248
Удаление учетных записей 249
Команда rmuser 249
Удаление записей из файла /etc/master.passwd 250
Удаление ничейных файлов 250
Работа с группами 251
Создание групп с помощью программы sysinstall 251
Редактирование файла /etc/group 252
Безопасность учетных записей 252
Выбор типа группы 252
Приватные группы 252
Группы совместных проектов 253
Смешанные группы 253
Задание стандартных прав доступа к файлам 253
Резюме 254
Глава 11. Инсталляция программного обеспечения 255
Виды программных пакетов 256
Исходный и исполняемый код 256
Способы распространения 257
Поиск программного обеспечения для FreeBSD 258
Использование стандартного системного ПО 258
Поиск ПО общего назначения для UNIX 259
Поиск коммерческого ПО 259
Инсталляция пакетов 260
Использование программы sysinstall 260
Использование специализированных утилит 262
Запрос информации о пакете 263
Коллекция переносимых приложений для FreeBSD 264
Инсталляция и обновление коллекции 265
Компиляция и инсталляция переносимого приложения 267
Удаление и обновление ПО 268
Удаление программы 268
Обновление программы 269
Проверка аутентичности программ 270
Резюме 271

Глава 12. Конфигурирование ядра 272
Общие сведения о ядре 273
Роль ядра во FreeBSD 273
Инсталляция и модификация ядра 274
Параметры загрузки ядра 275
Загрузка модулей ядра 275
Перекомпиляция ядра 276
Получение исходного кода ядра 276
Настройка ядра 277
Компиляция 278
Подключение нового ядра 279
Обновление ядра 279
Когда нужно обновлять ядро 279
Инсталляция исходных кодов 280
Компиляция и подключение нового ядра 280
Резюме 280
Глава 13. X Window System 281
Основы X-среды 282
Графическая модель 282
X-серверы 282
Наборы элементов управления 283
Диспетчеры окон 283
Графические оболочки 285
Другие компоненты 286
Сетевая поддержка 286
Конфигурирование X Window System 287
Предварительные замечания 287
Базовая структура файла XF86Config 287
Задание дисплея 289
Настройки графической платы 290
Настройки монитора 291
Настройки мыши 292
Добавление и удаление шрифтов 292
Конфигурационные утилиты 294
Пользовательские настройки 296
Выбор диспетчера окон и графической оболочки 296
Скорость перемещения указателя мыши 296
Скорость повтора 297
Запуск нескольких X сеансов 298
Резюме 299
Часть III. Сетевое конфигурирование 301
Глава 14. Основы сетевого конфигурирования 302
Локальные и коммутируемые сети 303
Сетевое оборудование 303
Протоколы TCP/IP 304
Другие сетевые стеки 305
Маршрутизация 306
IP-адреса и доменные имена 307
Использование статических IP-адресов 308
Получение IP-адреса 308
Настройка сетевых параметров с помощью программы sysinstall 309

Настройка адресов вручную 310
Активизация интерфейса 310
Настройка маршрутизации 311
Задание имени компьютера 312
Настройка DNS 312
Закрепление изменений 313
Протокол DHCP 314
Тестирование конфигурации 314
Базовая проверка с помощью команды ping 315
Проверка преобразования имен 316
Выявление ошибок маршрутизации 316
Резюме 317
Глава 15. Коммутируемые сети 318
Тестирование модема 319
Использование графического номеронабирателя 320
Выбор номеронабирателя 320
Создание соединения с провайдером 321
Создание файла аутентификации 321
Конфигурирование программы X ISP 322
Установление соединения 324
Использование текстовых утилит набора номера 325
Вспомогательные файлы 325
Конфигурирование протокола PPP 325
Файл /etc/ppp/options 325
Сценарий набора номера 327
Управление соединением 328
Применение протокола PPPoE в сетях DSL 328
Назначение протокола PPPoE 329
Программы для работы по протоколу PPPoE 329
Конфигурирование протокола PPPoE 329
Управление соединением 330
Резюме 330
Глава 16. Сетевые клиенты и серверы 331
Задачи клиентов и серверов 332
Порты и их применение 333
Номера портов 333
Стандартное распределение портов 334
Порты TCP, UDP и других протоколов 334
Структура сетевого соединения 335
Установление соединения 335
Маршрутизация 336
Ответ сервера 336
Брандмауэры и система NAT 337
Способы запуска серверов 338
Запуск вручную 338
Сценарии запуска 338
Использование суперсервера 339
Сетевая диагностика: утилита netstat 341
Получение информации об интерфейсах 341
Сбор статистических данных 342
Практическое применение утилиты 342
Резюме 342

Глава 17. Брандмауэры 343
Источники угрозы 344
Попытки взлома извне 344
Внутренние враги 346
Несанкционированное программное обеспечение во внутренней сети 346
Типы брандмауэров 347
Фильтры пакетов 347
Маршрутизаторы NAT 348
Прокси-серверы 349
Средства настройки брандмауэров во FreeBSD 349
Средства фильтрации пакетов: ipfw и IP Filter 349
Распространенные прокси-серверы 350
Создание сценария фильтрации 351
Необходимые параметры ядра 351
Стандартная политика фильтрации 352
Что необходимо разрешать, а что — запрещать 353
Создание правил 354
Синтаксис вызова утилиты ipfw 354
Инструкции утилиты ipfw 355
Синтаксис адресов утилиты ipfw 355
Примеры правил 356
Автоматизированная обработка сценария фильтрации 357
Конфигурирование NAT 358
Резюме 359

Часть IV. Серверы 361

Глава 18. Файловые серверы 362
Типы файловых серверов 363
Серверы доставки файлов и серверы совместного доступа к файлам 363
Дополнительные возможности файловых серверов 364
Распространенные файловые серверы и их применение 364
FTP-сервер 366
Распространенные FTP серверы 366
Запуск FTP-сервера 367
NFS-сервер 367
Определение экспортируемых файловых систем 368
Вопросы безопасности 369
Запуск NFS-сервера 370
Samba 370
Настройка глобальных параметров 370
Настройка паролей 372
Определение каталогов совместного доступа 372
Запуск сервера Samba 374
Резюме 375

Глава 19. Почтовые серверы 376
Типы почтовых серверов 377
Серверы принудительной доставки почты и серверы доставки по запросу 377
Серверы принудительной доставки почты 379
Серверы доставки почты по запросу 380
Конфигурирование сервера SMTP 380
Конфигурирование домена 381
Конфигурационные файлы sendmail 381

Изменение стандартных параметров 381
Задание компьютеров для приема почты 381
Задание шлюза исходящей почты 382
Борьба со спамом 383
Блокирование входящего спама 383
Блокирование исходящего спама 386
Конфигурирование сервера POP или IMAP 387
Конфигурирование программы Fetchmail 388
Конфигурационный файл 388
Запуск с помощью демона cron 390
Запуск в режиме демона 391
Резюме 391
Глава 20. Web-серверы 392
Когда нужно запускать Web сервер 393
Web-серверы для FreeBSD 394
Основы конфигурирования сервера Apache 394
Запуск Apache 395
Конфигурационный файл 395
Базовые параметры сервера 396
Параметры серверных каталогов 397
Модули Apache 398
Создание Web-страниц 399
Структура HTML-документа 400
Средства создания Web-страниц 402
Создание переносимых Web-страниц 403
Резюме 404
Глава 21. Серверы дистанционной регистрации 405
Типы серверов дистанционной регистрации 406
Конфигурирование сервера Telnet 407
Схема функционирования протокола 407
Запуск сервера Telnet 408
Ограничение доступа к серверу 408
Конфигурирование сервера SSH 409
Преимущества SSH 409
Выбор сервера SSH 410
Запуск сервера SSH 411
Настройка параметров сервера 411
Настройка удаленного доступа в X Window System 413
Взаимодействие клиента и сервера X Window System 413
Регистрация в X-среде в текстовом режиме 414
Базовая процедура 414
Туннелирование X трафика средствами SSH 415
Применение утилиты xauth 416
Регистрация по протоколу XDMCP 417
Конфигурирование сервера XDMCP 417
Использование клиента XDMCP 419
Использование FreeBSD-системы в качестве выделенного X-терминала 420
Запуск сервера VNC 421
Взаимодействие клиента и сервера VNC 422
Инсталляция VNC 422
Конфигурирование сервера VNC 423

Пользовательские модификации 423
Модификация сценария vncserver 424
Установление VNC-соединения 425
Запуск VNC под управлением демона inetd 426
Резюме 428
Глава 22. Вспомогательные серверы 429
Серверы DHCP 430
Когда запускать DHCP-сервер 430
Инсталляция DHCP-сервера 431
Основы конфигурирования 432
Задание глобальных параметров 432
Определение подсети 433
Назначение статических IP адресов 433
Серверы DNS 434
Когда запускать DNS-сервер 435
Базовое конфигурирование сервера 436
Конфигурирование зоны 438
Настройка прямого преобразования 438
Настройка обратного преобразования 439
Серверы синхронизации времени 440
Протокол NTP 440
Поиск родительских серверов 441
Конфигурирование протокола NTP 442
Настройка таймера на этапе начальной загрузки 442
Конфигурирование NTP-сервера 443
Серверы шрифтов 444
Обработка шрифтов в X-среде 444
Конфигурирование сервера шрифтов 445
Создание каталогов шрифтов 445
Изменение конфигурационного файла 445
Запуск сервера шрифтов 446
Использование сервера шрифтов 446
Резюме 447
Часть V. Пользовательское программное обеспечение 449
Глава 23. Настольные графические оболочки 450
Роль графических оболочек 451
Обзор доступных графических оболочек 453
KDE 453
GNOME 455
XFce 456
Запуск собственной графической оболочки 457
Работа в среде GNOME 459
Запуск GNOME 459
Операции с файлами 460
Конфигурирование рабочего стола 460
Использование программы Control Center 460
Установка привязок между типами файлов и программами-обработчиками для диспетчера Nautilus 461
Модификация внешнего вида программ 464
Изменение настроек клавиатуры и мыши 465
Добавление программ в экранные меню 465
Резюме 467

Глава 24. Сетевые утилиты **468**
Почтовые клиенты 469
Наиболее распространенные клиенты 469
Начальное конфигурирование 470
Чтение и хранение электронной почты 472
Отправка электронной почты 473
Вложения 474
Web-броузеры 475
Наиболее распространенные броузеры 475
Конфигурирование броузера 476
Безопасность броузера 478
Клиенты FTP и SFTP 479
Наиболее распространенные FTP клиенты 480
Базовые FTP-команды 481
Применение FTP: передача файлов 482
Утилита ftp 482
Программа gFTP 484
Клиенты совместного доступа к файлам 484
Экспортируемые каталоги NFS 484
Точки совместного доступа SMB/CIFS 486
Утилита smbclient 486
Команда mount_smbfs 487
Клиенты дистанционной регистрации 489
Клиенты Telnet 489
Клиенты SSH 489
Резюме 490
Глава 25. Офисные средства **491**
Офисные пакеты FreeBSD 492
GNOME Office 492
KOffice 493
OpenOffice.org 494
Вспомогательные средства 495
Особенности офисных пакетов 496
Обработка шрифтов 496
Печать 497
Импорт и экспорт файлов 498
Образец офисного пакета: OpenOffice.org 499
Конфигурирование шрифтов и принтеров 499
Добавление принтеров 499
Добавление шрифтов 501
Создание документа 502
Редактирование текста 502
Табличные вычисления 504
Создание векторной графики 506
Создание презентации 507
Резюме 509
Глава 26. Графические средства **510**
Программа GIMP 511
Запуск программы 511
Обзор возможностей 512
Загрузка и сохранение графических изображений 513
Создание текстовых и графических элементов 514

Выбор цветов и шаблонов заливки 514
Инструменты рисования 515
Заливка 515
Добавление текстовых надписей 516
Преобразования 516
Выборка и усечение изображений 517
Изменение цветов 517
Применение фильтров 517
Программы векторной графики 517
Xfig 518
Dia 519
Gnuplot 520
Обработка файлов PostScript, EPS и PDF 523
Программа Ghostscript 524
Графические утилиты просмотра PostScript-файлов 525
Графические утилиты просмотра PDF-файлов 526
Резюме 527

Глава 27. Мультимедийные и игровые приложения 528
Обработка звука во FreeBSD 529
Поддерживаемое звуковое оборудование 529
Запуск звуковой подсистемы 530
Программы работы со звуком 532
Микшеры 532
Текстовые утилиты записи и воспроизведения звука 533
Графические утилиты записи и воспроизведения звука 534
Программы для работы с MP3 файлами 535
Создание MP3-файлов 536
Воспроизведение MP3-файлов 537
Проигрыватели музыкальных компакт дисков 537
Мультимедийные проигрыватели 538
Форматы мультимедийных файлов 539
Воспроизведение мультимедийных файлов 540
Подключение проигрывателя к броузеру 541
Игры 543
Текстовые игры UNIX 543
Игры для X Window System 544
Коммерческие игры 545
Резюме 545

Часть VI. Обслуживание системы 547

Глава 28. Автоматизированные и неавтоматазированные процедуры обслуживания 548
Очистка файлов 549
Журнальные файлы 549
Временные файлы 551
Неиспользуемые программы 552
Учет пользовательских файлов 552
Контроль загруженности центрального процессора 554
Утилиты для контроля процессора 554
Выявление случаев чрезмерной загруженности процессора 556
Устранение проблем 556
Предотвращение проблем 557
Контроль использования памяти 558

Утилиты для контроля памяти 558
Определение доступной памяти 558
Определение потребности в памяти 559
Выявление случаев чрезмерной загруженности памяти 559
Предотвращение и устранение проблем 560
Увеличение области подкачки 560
Добавление раздела подкачки 560
Добавление файла подкачки 561
Обновление программного обеспечения 562
Важность обновлений 562
Средства и процедуры проверки обновлений 562
Какие программы обновлять 563
Пакетные обновления 563
Обновления в стиле make world 565
Переинсталляция FreeBSD 567
Резюме 567
Глава 29. Системная безопасность 568
Способы атак 569
Зондирование системы 569
Дистанционные атаки 570
Локальные атаки 571
Непрямые атаки 572
Атаки типа "отказ от обслуживания" 573
Удаление ненужного программного обеспечения 574
Выявление неиспользуемых программ 574
Ограничение доступа к программам 575
Отключение программ 576
Удаление программ 577
Возможности преодоления ограничений 577
Ограничение доступа к серверам 578
Парольная защита серверов 578
Специфические средства защиты серверов 579
Программа TCP Wrappers 580
Применение брандмауэра 582
Безопасность паролей 583
Важность защиты паролей 583
Создание паролей 584
Замечания по поводу учетной записи root 585
Обнаружение взлома 586
Общие признаки 586
Программа Tripwire 587
Принципы функционирования 587
Инсталляция 588
Изменение конфигурации 588
Инициализация и обновление базы данных 589
Применение 590
Другие программы обнаружения взлома 590
Что делать при выявлении взлома 591
Резюме 593
Глава 30. Компиляция программ 594
Принципы компиляции 595
Процесс компиляции 595

За и против 596
Средства компиляции 597
Компиляторы машинных языков 597
Вспомогательные библиотеки и файлы заголовков 597
Универсальные системы разработки 598
Стандартная процедура компиляции 599
Получение программы 599
Чтение документации 600
Конфигурирование программы 600
Компиляция программы 601
Инсталляция скомпилированных программ 602
Выявление конфликтующих пакетов 602
Процесс инсталляции 603
Проверка полученного результата 604
Резюме 604

Глава 31. Написание сценариев 605
Компилируемые и интерпретируемые программы 606
Машинный и текстовый код 606
Интерпретаторы 607
За и против 608
Распространенные языки сценариев 608
Запуск и останов сценария 609
Использование внешних команд 610
Выполнение внешних команд 610
Каналы и переадресация потоков ввода вывода 611
Переменные 612
Присвоение значений переменным 612
Использование аргументов командной строки в качестве переменных 613
Применение переменных 613
Отображение значения переменной 613
Прием данных от пользователя 614
Модификация строковых переменных 614
Модификация числовых переменных 615
Условные выражения 616
Основные логические операции 616
Инструкции if, elif, else 617
Инструкция case 619
Условное выполнение команд 620
Циклы 620
Циклы for 620
Циклы while 621
Циклы until 622
Функции 623
Резюме 625

Глава 32. Поиск и устранение неполадок 626
Получение справки 627
Справочные ресурсы FreeBSD 627
Печатная документация 628
Интерактивные источники информации 629
Выявление аппаратных и программных проблем 630
Распространенные признаки аппаратных проблем 631
Распространенные признаки программных проблем 632

Локализация проблемы 633
Общие процедуры локализации 633
Аппаратная диагностика 634
Проблемы, связанные с процессором 634
Проблемы, связанные с материнской платой 635
Проблемы, связанные с памятью 636
Проблемы, связанные с дисками 636
Проблемы, связанные с видеоподсистемой 637
Проблемы, связанные со звуковой платой 638
Проблемы, связанные с внешними устройствами ввода вывода 638
Проблемы, связанные с сетевыми устройствами 639
Проблемы, связанные с принтером 639
Программная диагностика 640
Проблемы, связанные с запуском системы 640
Проблемы, связанные с драйверами устройств 641
Проблемы, связанные с сетевым обеспечением 641
Проблемы, связанные с пользовательскими программами 642
Тестирование полученного результата 643
Использование аварийной системы 643
Резюме 645
Словарь терминов 646
Предметный указатель 664

Об авторе

Родерик Смит — известный автор книг по компьютерной тематике. Является пользователем UNIX с 1987 года; администрированием UNIX-систем, включая FreeBSD занимается с 1994 года. *FreeBSD: The Complete Reference* — его десятая книга. Среди других написанных им книг стоит особо отметить *Advanced Linux Networking* (Addison Wesley, 2002) и *The Multi-Boot Configuration Handbook* (Que, 2000).

Благодарности

Книгу помогает создавать множество людей чьи имена не указываются на обложке. В первую очередь я хотел бы поблагодарить сотрудников издательства McGraw-Hill/Osborne, внесших огромный вклад в работу над книгой Франни Келли (Franny Kelly) давала очень помогавшие мне указания Эмма Акер (Emma Acker) и Мартин Пржибила (Martin Przybyla) решали организационные вопросы. Лора Стоун (Laura Stone), редактор проекта, контролзировала процесс редактирования Марти Пол (Marty Paul), литературный редактор, исправлял грамматические ошибки и указывал мне на неграмотно сформулированные мысли. Гленн Бекер (Glenn Becker) был техническим редактором книги и не давал мне сбиться с пути. Оставшиеся ошибки, естественно, на моей совести. Викки Хардинг (Vicki Harding) и Шерри Рогельберг (Sherry Rogelberg) из компании Studio B Productions помогли мне наладить контакты с издательством McGraw-Hill/Osborne и запустить книгу в производство Выражаю так же признательность Дэвиду Кингу (David King), который обсуждал со мной тонкости операционной системы FreeBSD И наконец, нельзя не поблагодарить все сообщество пользователей FreeBSD: без их преданности операционная система не существовала бы

Введение

На рынке настольных систем и серверов нижнего класса доминирует несколько групп операционных систем. Наиболее широко распространены продукты компании Microsoft: Windows 2000 и Windows XP. Для определенного типа оборудования важную роль играет Mac OS. Кроме них, имеется класс операционных систем с *открытым исходным кодом*, которые разрабатываются отдельными специалистами или организациями и свободно распространяются среди желающих. Результатом такой деятельности стало появление довольно мощных систем, производных от UNIX — чрезвычайно популярной операционной системы, которая возникла в конце 1960-х и в том или ином виде сохранилась до наших дней. Многие элементы вычислительных систем, принимаемые сегодня как данность (достаточно назвать протоколы TCP/IP), берут начало именно в UNIX. Особенно популярны два семейства UNIX-систем: BSD и Linux. Эта книга посвящена самому популярному члену первого семейства, а именно FreeBSD. На тот момент, когда писалась книга (начало 2003 года), самая последняя версия системы имела номер 5.0.

Порой FreeBSD ошибочно считают разновидностью Linux, но это не так, несмотря на то что FreeBSD и Linux имеют много общих характеристик и зачастую одинаковое программное обеспечение. Linux строится вокруг *ядра* операционной системы (ядро служит своего рода мостом между пользовательскими программами и аппаратными средствами), которое представляет собой переработанное "с нуля" ядро UNIX. В то же время возникновение FreeBSD связано с решением переписать ядро UNIX на по файловой основе (подробнее об этом рассказывается в главе 1). Важно также отметить, что во FreeBSD и Linux используется разный набор низкоуровневых утилит и файлов конфигурации. Схожие различия существуют между FreeBSD и коммерческими разновидностями UNIX, такими как AIX компании IBM, Solaris компании Sun и HP-UX компании Hewlett-Packard. Поэтому даже тем, кто знаком с Linux или какой-нибудь коммерческой версией UNIX, придется потратить время на изучение особенностей FreeBSD. Для новичков важно иметь руководство именно по FreeBSD, а не другой, пусть и родственной, операционной системе. Здесь-то и заключается главное достоинство данной книги, которая посвящена исключительно FreeBSD.

Для кого предназначена эта книга

FreeBSD — сложная и многоцелевая операционная система. Ее можно использовать как дома (электронная почта, Internet, игры и т.д.), так и в офисе (обработка текста и электронных таблиц, другие офисные применения), на рабочей станции ученого или инженера (сложные программы анализа данных) и на сетевом сервере (Apache, Samba, `sendmail` и другие популярные серверные приложения, доставляющие информацию клиентам в пределах одного офиса или по всему миру). Компьютер, на котором установлена FreeBSD, может играть роль выделенного маршрутизатора или брандмауэра, защищающего локальную сеть от несанкционированного доступа. Широкий диапазон возможностей применения FreeBSD и обусловил объем книги, которая претендует на то, чтобы называться полным руководством. В большей или меньшей степени книга охватывает все перечисленные выше аспекты применения FreeBSD.

Диапазон возможностей FreeBSD также означает, что эта книга предназначена для широкого круга читателей. При ее написании автор старался не делать никаких пред

положений насчет уровня знаний читателей т е не предполагается предварительное знакомство либо опыт работы с FreeBSD или какой-нибудь другой разновидностью UNIX, хотя наличие определенной подготовки, разумеется, не помешает Читатель может быть профессиональным системным администратором, мало знакомым с возможностями FreeBSD или вообще UNIX, — например, администратором Windows, решившим освоить FreeBSD Но точно так же книга может попасть в руки пользователя любителя, которому интересно поэкспериментировать с FreeBSD. Невозможно предугадать, сколько пользователей работают в системе — один или сто — и каковы их потребности Но наверняка каждый читатель найдет в книге "свои" главы, информация в которых заинтересует его больше всего

В книге делается упор на администрирование FreeBSD, а не на обычное использование системы Такое разделение четко прослеживается именно во FreeBSD и гораздо менее – в других операционных системах, включая Windows 9x/Me, Mac OS Classic и даже самые последние разработки компаний Microsoft и Apple (Windows NT/2000/XT и Mac OS X). Во FreeBSD (и вообще в UNIX) компьютер конфигурирует системный администратор. он настраивает сетевые параметры, инсталлирует и удаляет программы, а также создает учетные записи для остальных пользователей, которые, собственно, и работают за компьютером — пишут программы, проводят вычисления, занимаются графикой и т.д. Что касается сервера, то системный администратор может быть единственным его пользователем с обычной учетной записью; другие пользователи подключаются к компьютеру дистанционно, иногда в анонимном режиме Если читатель располагает учетной записью во FreeBSD, но не является системным администратором этого компьютера, книга будет ему полезна не в полном объеме, хотя некоторая информация, особенно в части V, может очень пригодиться. Руководство ориентировано в основном на администраторов FreeBSD-систем но будет полезно и пользователям устанавливающим FreeBSD для личных целей.

Структура книги

Материал книги разделен на шесть частей каждая из которых состоит из 4 9 глав (всего 32 главы)

- **Часть I, "Инсталляция FreeBSD"**, содержит четыре вводные главы, посвященные анализу системных требований, инсталляции операционной системы краткому обзору возможностей инсталлированной системы и использованию FreeBSD на компьютере с несколькими операционными системами Читателям, устанавливающим FreeBSD "с нуля", эти главы будут особенно интересны. Если же на компьютере уже установлена FreeBSD, вводную часть можно пропустить. Впрочем, тем читателям, которым никогда прежде не приходилось иметь дело с UNIX, желательно прочесть главу 3 Глава 4 будет очень полезна для тех, кто работает со съемными дисками или выполняет программы с использованием эмулятора
- **Часть II "Основы системного администрирования"**, состоит из девяти глав – она самая объемная в книге. Здесь рассматриваются все основные задачи администратора FreeBSD. запуск FreeBSD, выполнение программ, управление разделами диска и файлами, конфигурирование принтеров, управление учетными записями пользователей, установка программного обеспечения, конфигурирование ядра и X Window System (графической среды FreeBSD) Большинство системных администраторов FreeBSD рано или поздно столкнутся со всеми этими задачами (разве что X-среду не так часто приходится настраивать)

- **Часть III "Сетевое конфигурирование"**, содержит четыре главы, посвященные конфигурированию сети, настройке удаленного доступа, принципам функционирования клиентов и серверов, а также конфигурированию межсетевых фильтров (брандмауэров) Семейство протоколов TCP/IP было разработано для UNIX-систем, поэтому неудивительно, что средства сетевого конфигурирования FreeBSD тесно связаны с этой операционной системой. Таким образом сетевое конфигурирование — лишь особый вид базовой процедуры системного администрирования
- **Часть IV, "Серверы"**, состоит из пяти глав, в которых описаны ключевые серверные приложения, способные работать под управлением FreeBSD: файловые серверы, серверы электронной почты, Web-серверы, серверы регистрации пользователей и различные вспомогательные серверы. Поскольку FreeBSD-системы часто используются в качестве серверов, эта часть книги особо значима для администраторов FreeBSD В дополнение к изложенной здесь информации читателям следует обратиться к другим источникам, посвященным конкретному серверу например Web-серверу Apache или файловому серверу Samba
- **Часть V, "Пользовательское программное обеспечение"**, посвящена установке и конфигурированию пользовательских программ. В пяти главах этой части приведена информация о настольных графических оболочках, сетевых клиентах офисных пакетах графических редакторах и мультимедийных приложениях (включая игры)
- **Часть VI, "Обслуживание системы"** содержит пять глав, в которых описаны процедуры автоматизированного и неавтоматизированного регламентного обслуживания, рассмотрены вопросы безопасности, компиляции программ, написания сценариев и устранения неполадок

Главы можно читать по порядку, хотя это необязательно. В основном главы не зависят друг от друга однако некоторые главы опираются на информацию, содержащуюся в других разделах книги. Читатели которые уже знакомы (хотя бы в общих чертах) с FreeBSD или другими разновидностями UNIX, могут пропустить вводный материал, который покажется им излишним. Если нужно быстро найти информацию по интересующему вопросу или уточнить значение незнакомого термина воспользуйтесь предметным указателем и словарем терминов

Типографские особенности книги

В книге соблюдаются определенные правила, которые способствуют более качественному усвоению материала и позволяют разбивать страницы книги на информационные блоки, облегчающие ее просмотр и поиск нужного материала К типографским особенностям относятся шрифтовые выделения отдельных слов и строк, а также специальные элементы форматирования, применяемые к целым абзацам Особым образом выделяются также IP-адреса и доменные имена компьютеров

Шрифтовые выделения

В книгах компьютерной тематики некоторые слова могут быть истолкованы неоднозначно, поскольку представляют собой названия программ, команд, компьютеров и т.д. При описании диалога между человеком и компьютером не всегда понятно, что вводится человеком, а что отображается на экране монитора В связи с этим в книге применяются следующие шрифтовые выделения, проясняющие ситуацию

- Обычный текст записывается с использованием стандартного пропорционального шрифта (как этот)
- **Полужирным шрифтом** выделяются начальные элементы маркированных и нумерованных списков в качестве своеобразных заголовков.
- *Курсивный шрифт* обозначает новый термин, который в данной главе упоминается впервые. Кроме того, курсив применяется для привлечения внимания читателя к выделенному тексту.
- `Моноширинным шрифтом` записываются имена файлов и компьютеров, URL адреса, синтаксические конструкции, фрагменты конфигурационных файлов, а также текст, отображаемый на экране компьютера. Этим же шрифтом иногда выделяются названия программ, особенно в тех случаях, когда программа известна по имени исполняемого файла.
- **`Полужирный моноширинный шрифт`** используется для обозначения текста, вводимого пользователем. Как правило, это происходит по приглашению интерпретатора команд, т.е. в командной строке, но иногда текст вводится при взаимодействии с программой без участия интерпретатора команд.
- *`Моноширинный курсивный шрифт`* применяется для обозначения переменных, т.е. информации, которая меняется в разных системах или ситуациях. Например, ссылка на файл может выглядеть так: *`файл`*`.txt`. Эта запись говорит о том, что имя файла может быть любым или же на данный момент не известно.
- Особым шрифтом выделяются названия элементов оконных интерфейсов.

Работу в текстовом режиме контролирует *интерпретатор команд*, который представляет собой программу, предназначенную для приема команд от пользователя. Во FreeBSD интерпретатор запускается автоматически после регистрации пользователя в системе. Длинные команды или примеры интерактивного взаимодействия располагаются отдельно от основного текста. В таких случаях указывается *приглашение интерпретатора команд*, т.е. символ, которым интерпретатор обозначает готовность к приему данных. В качестве такого приглашения для учетной записи суперпользователя (`root`) используется знак решетки (#), а для учетных записей рядовых пользователей — знак доллара ($). Например, строка

```
$ ifconfig | grep vr0
```

означает, что команда выполняется рядовым пользователем (в большинстве случаев пользователь `root` может ввести аналогичную команду, но для этого не обязательно обладать правами суперпользователя). Если строка начинается со знака #, значит команда будет правильно работать только в режиме суперпользователя. Для команд, приводимых в самом тексте, это правило не соблюдается во избежание путаницы. Например, может быть написано следующее: "Для просмотра информации о сетевых интерфейсах введите команду `ifconfig`".

Некоторые строки в листингах могут не помещаться по ширине книжной страницы. В таких случаях первая строка будет оканчиваться специальной стрелкой, обозначающей продолжение строки, например:

```
/usr/src -network 172.17.2  netmask 255.255.255.0 nova nebula ↳
blackhole browndwarf
```

Если присутствует символ ↳, значит, приведенный текст необходимо рассматривать как одну строку. В то же время во многих файлах конфигурации FreeBSD разрешено в качестве символа продолжения строки использовать обратную косую черту ('\'). Это средство, впрочем, не является универсальным, поэтому в случае сомнений следует работать с длинной строкой.

Специальные элементы форматирования

Для выделения важной информации, на которую читателям стоит обратить особое внимание, используется следующий набор элементов форматирования.

В виде примечаний приводится информация, которая, с точки зрения автора, должна быть интересной или полезной читателю, но несколько второстепенна по отношению к основной теме излагаемого материала К примеру, в примечании может быть указано, чем та или иная команда отличается от своих аналогов в других операционных системах.

В качестве совета приводится практический пример или команда, которые помогут сэкономить время при решении поставленной задачи или достичь цели несколько необычным (или неочевидным), но весьма эффективным способом.

Цель предупреждения — уведомить читателя о потенциальной опасности. Эта опасность может быть связана с некорректной конфигурацией системы, синтаксической ошибкой и даже неприятностями с законом, к которым приводят выполняемые действия. Всегда обращайте внимание на предупреждения

Врезка

Врезка обычно больше, чем примечание, совет и предупреждение и состоит из нескольких абзацев. Ее можно считать расширенным примечанием

Обозначение IP-адресов и имен компьютеров

Те, кто знаком с протоколами TCP/IP, очевидно, знают, что определенные IP адреса зарезервированы для *частных* сетей, т.е. сетей, которые изолированы от Internet Это, в частности, аяреса 10 0 0 0—10.255 255.255, 172.16.0 0- 172.31.255.255 и 192 168 0 0 192 168 255 255. В большинстве примеров книги, требующих указания IP адресов, используются именно зарезервированные алреса во избежание случайных совпадений с реальными IP адресами, причем даже для систем, которые в примере имеют выход в Internet.

Возможно, читателям также известно, что в одном из документов серии RFC а именно RFC 2606, описаны домены, имена которых зарезервированы, примерно как указанные выше IP-адреса частных сетей В частности, имена `example.com`, `example.net` и `example.org` зарезервированы для использования в документации И хотя в нескольких примерах книги эти имена все же участвуют, они уже слишком "приелись" Кроме того, во многих примерах приходится использовать сразу несколько доменных имен, а эти три зарезервированных имени слишком похожи одно на другое, что неизбежно вызывает путаницу Поэтому автором выбраны доменные имена, которые пока не зарегистрированы, например `luna.edu` `pangaea.edu` и `threeroomco.com`. Заранее приношу извинения на тот случай, если какие-либо из этих доменов станут реальными и введут кого-то в заблуждение В ряде примеров, впрочем, участвовали действительно существующие домены — это оговаривалось в тексте

О содержимом компакт-диска

К книге прилагается компакт-диск с операционной системой FreeBSD 5 1 FreeBSD — очень большая ОС, которая не в состоянии уместиться на одном компакт-диске, поэтому в дистрибутив включены только ядро системы и самые важные утилиты Для начального знакомства этого будет вполне достаточно FreeBSD можно инсталлировать, загрузившись непосредственно с компакт диска (или создав загрузочную дискету с помощью утилит, которые записаны на компакт-диске, как рассказывается в главе 2). Поскольку FreeBSD — система с открытым исходным кодом, ее разрешается инсталлировать на любом количестве компьютеров.

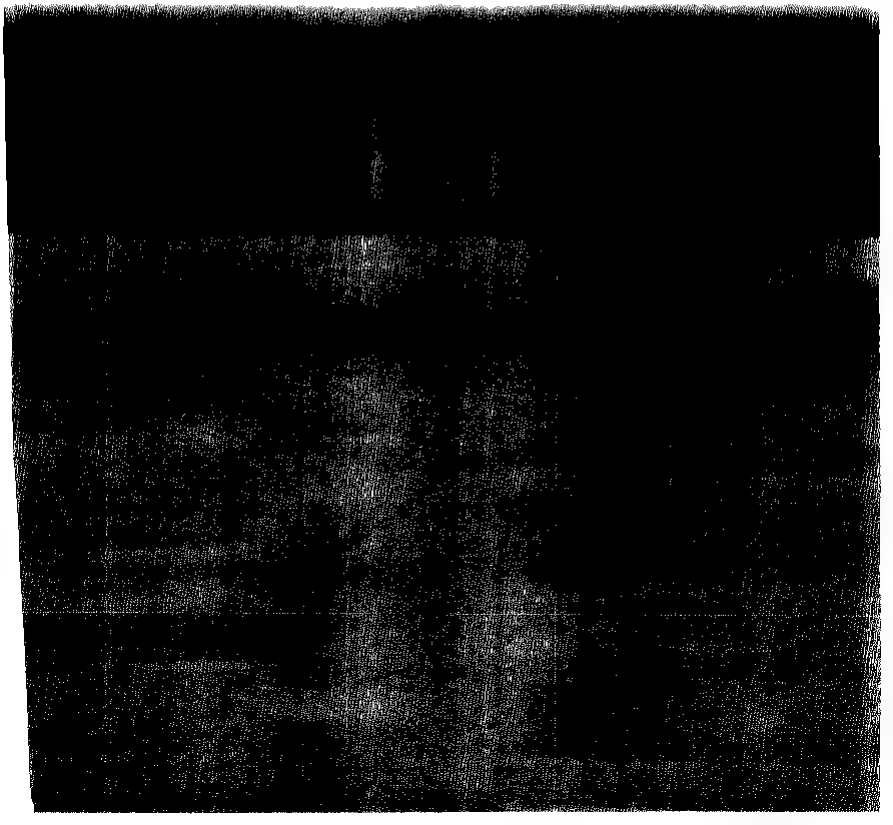
Если возникнет необходимость инсталлировать программные средства, описаниме в книге, но не включенные в прилагаемый компакт-диск, то их можно найти в Internet Проще всего это сделать, инсталлировав дополнительные пакеты с официального FTP-узла FreeBSD, как описано в главе 11 Можно также загрузить систему с компакт-диска, но саму FreeBSD инсталлировать с FTP-узла, что позволит сразу установить программы, которых нет на прилагаемом компакт-диске Сетевая инсталляция удобна при наличии высокоскоростного подключения к сети. Но если используется модем, скорость передачи которого составляет менее 56 Кбит/с, то на загрузку всех выбранных пакетов уйдет не один час В этом случае лучше сначала инсталлировать все необходимое с компакт диска а затем загрузить нужные пакеты из Internet

Как связаться с автором книги

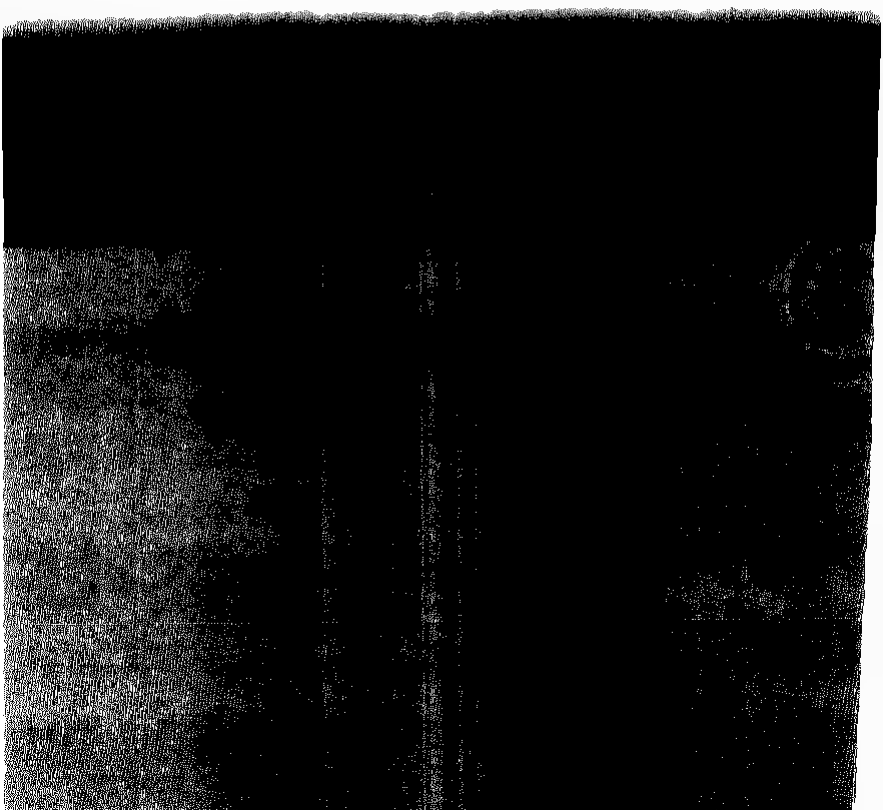
Если у читателей возникнут замечания или вопросы по поводу книги, то со мной можно связаться по адресу `rodsmith@rodsbooks.com` Я веду также Web-страницу посвященную этой книге. Ее адрес таков

```
http://www rodsbooks com/freebsd
```

Полный справочник по



Полный справочник по



Эта книга послужит читателям надежным руководством по операционной системе FreeBSD. Но прежде чем начинать знакомство с системой, нужно ответить на ряд важных вопросов: зачем устанавливать FreeBSD, какое программное обеспечение в ней работает и какую аппаратную платформу лучше всего использовать? Об этом и пойдет речь в настоящей главе. Понимание особенностей операционной системы позволит избежать дорогостоящих ошибок, таких, например, как попытка эксплуатации системы на неадекватном оборудовании или в неподходящей среде. Для новичков эта глава ключевая. Но даже читателям, уже знакомым с FreeBSD (или другими разновидностями UNIX), приведенная здесь информация поможет подготовиться к инсталляции системы. В главе 2 процесс инсталляции описан более подробно.

Для чего нужна FreeBSD

В любом деле важно находить правильные средства для достижения поставленных целей, и выбор оптимальной операционной системы (ОС) — не исключение. FreeBSD — очень гибкая и эффективная система, имеющая множество возможных вариантов применения. Но несмотря на это она не всегда оказывается наилучшим выбором. Вот почему нужно прежде всего проанализировать, в каких обстоятельствах следует инсталлировать именно FreeBSD.

FreeBSD устанавливают либо на рабочей станции, либо на сервере. Ниже рассматриваются достоинства и недостатки системы в каждом из этих случаев. Кроме того, FreeBSD — член семейства операционных систем, производных от (или смоделированных на основе) UNIX. Поэтому важно понимать, как FreeBSD (и UNIX в целом) соотносится с операционными системами других типов и какое место занимает FreeBSD в семействе UNIX-подобных систем.

FreeBSD в роли настольной ОС

В этой книге под термином *рабочая станция* понимается компьютер, за которым пользователь решает свои повседневные задачи: редактирует текст, программирует, занимается Web-дизайном (играет в игры...). Иногда термин употребляют в более узком смысле, подразумевая, что указанные выше функции выполняются на высокопроизводительных компьютерах или системах, постоянно подключенных к сети. В этом случае низкопроизводительные или изолированные системы называют *настольными компьютерами*. Впрочем, уровень вычислительных мощностей постоянно растет, соответственно меняется и оценка производительности. Что касается подключения к сети, то нужно учитывать огромную популярность коммутируемых сетевых соединений в системах нижнего сегмента рынка. Исходя из этого мы будем употреблять термин "рабочая станция" в отношении компьютеров конечных пользователей.

Способность компьютера играть роль рабочей станции зависит от двух основных факторов:

- **Удобный пользовательский интерфейс**. Современные пользователи привыкли к наличию удобных средств работы с компьютером. Ключевое из них — пользовательский графический интерфейс. Во FreeBSD графический интерфейс представлен средой *X Window System* (сокращенно — X-среда). Ее ядро довольно примитивно в сравнении с другими аналогичными интерфейсами, поэтому к нему добавлен ряд надстроек, таких как *диспетчер окон* (управляет рамками окон) и набор *элементов управления* (упрощают программисту задачу отображения меню, диалоговых окон и т.д.). Подобный модульный подход делает X среду очень гибкой с точки зрения программирования интерфейса, но есть и

обратная сторона медали: возникающая несогласованность между приложениями часто сбивает пользователей с толку.

- **Доступность прикладных программ** На рабочей станции выполняются пользовательские приложения: текстовые редакторы, программы обработки электронных таблиц, почтовые клиенты, компиляторы, графические редакторы и даже игры. Если операционная система не может предложить пользователю необходимый набор прикладных программ, то она не подходит для установки на рабочей станции. Список приложений, доступных во FreeBSD, велик, хотя и не настолько, как хотелось бы. В частности, популярный пакет Microsoft Office не перенесен во FreeBSD. Существуют его альтернативы, такие как пакет OpenOffice.org (описан в главе 25), но они не обеспечивают 100-процентную совместимость файлов. Если говорить в целом, то в одних категориях (например, игры) ощущается нехватка приложений для FreeBSD, а в других их более чем достаточно.

FreeBSD вполне подходит на роль операционной системы для рабочей станции, хотя в конкретной ситуации выбор зависит от специфических потребностей пользователя. К достоинствам FreeBSD относятся низкая стоимость, способность функционировать на старом оборудовании, стабильность, гибкий пользовательский интерфейс и близкое сходство с другими разновидностями UNIX. К недостаткам можно отнести менее согласованный, чем в Windows или Mac OS, пользовательский интерфейс, несколько непривычный для пользователей вышеуказанных систем. Что касается фактора доступности прикладных программ, то все зависит от обстоятельств. Если пользователь вынужден работать с приложениями вроде Microsoft Office, которые недоступны во FreeBSD, придется обратиться к другой операционной системе. Если же пользователь не привязан к конкретным форматам файлов, то можно сделать выбор в пользу FreeBSD.

Если на рабочей станции требуется установить UNIX-подобную операционную систему и при этом предполагается запускать Microsoft Office или другой прикладной пакет, недоступный во FreeBSD, то в качестве компромиссного варианта подойдет Mac OS X. Эта система создана на основе микроядра Mach и целого ряда компонентов FreeBSD. Конечно, Mac OS X — это не FreeBSD, но благодаря многочисленным "заимствованиям" из FreeBSD и других разновидностей UNIX она гораздо больше знакома администраторам UNIX, чем Windows-системы или более ранние версии Mac OS. В Mac OS X поддерживаются многие прикладные пакеты, включая Microsoft Office.

Читатели, установившие FreeBSD на рабочей станции, должны внимательно прочитать часть V. Если к тому же предполагается, что в системе будет работать большое количество пользователей, то нужно уделить особое внимание вопросам конфигурирования графических оболочек (глава 23). Эти оболочки значительно усовершенствовались с середины 90-х и по-прежнему регулярно обновляются.

FreeBSD в роли серверной ОС

Серверы — это компьютеры с сетевыми платами, обрабатывающие запросы других компьютеров. Именно серверы обеспечивают наиболее эффективное функционирование сетей. Ниже рассмотрены примеры серверов.

- **Файловый сервер** позволяет другим компьютерам читать и записывать файлы серверной файловой системы. Эти серверы можно использовать для распространения свободно доступных файлов (например, компонентов FreeBSD), а также в качестве централизованного хранилища файлов локальной сети.

- **Web-сервер** В определенном смысле это разновидность файлового сервера Обычно служит для доставки файлов Web клиенту, называемому *броузером*
- **Почтовый сервер** – это компьютер, обменивающийся почтовыми сообщениями с другими серверами либо с почтовыми клиентами конечных пользователей. В Internet обмен почтой происходит в основном между серверами, тогда как в локальной сети достаточно одного почтового сервера
- **Сервер дистанционной регистрации** FreeBSD можно сконфигурировать на прием регистрационных запросов от других пользователей. В этом случае работа в системе ведется в дистанционном режиме, т е. стирается грань между рабочей станцией и сервером (система реализует функции сервера, но используется в основном как рабочая станция) Иногда на сервере предусматривается возможность дистанционной регистрации администратора
- **Вспомогательные серверы** выполняют служебные функции Например, сервер DHCP (Dynamic Host Configuration Protocol — протокол динамического конфигурирования компьютеров) позволяет централизованно задавать параметры сетевой конфигурации клиентских компьютеров Сервер шрифтов предоставляет рабочим станциям информацию о шрифтах

Особенности различных серверов подробно описаны в части IV. FreeBSD является прекрасным кандидатом на роль серверной ОС Не в последнюю очередь это связано с тем, что многие популярные серверные программы изначально разрабатывались в среде UNIX и потому прекрасно работают во FreeBSD В сочетании с недорогими аппаратными компонентами платформы x86 система на базе FreeBSD является весьма экономичным решением для большинства современных серверных программ

На заметку

Термин "сервер" может обозначать как компьютер, так и программу, реализующую серверные функции Это означает, что один компьютер способен быть сервером сразу нескольких типов, например почтовым и Web-сервером В большинстве случаев смысл термина "сервер" ясен из контекста

Преимуществом использования FreeBSD в качестве серверной ОС является надежность системы Несмотря на то что компьютеры, работающие на базе процессоров x86, весьма разнородны и не всегда столь же надежны, как традиционные серверные компьютеры, выпускаемые, к примеру, компаниями Sun и Silicon Graphics, FreeBSD вполне стабильна Нередко серверы FreeBSD работают без сбоев месяцами Если происходит перезагрузка, то это чаще делается для обновления аппаратных, а не программных компонентов Перезагружать зависшую или нестабильно работающую систему приходится еще реже Поскольку от сервера зависит работоспособность множества компьютеров, высокая надежность — очень важное качество операционной системы

Серверным компьютерам требуется эффективное и надежное сетевое программное обеспечение Сюда входят как отдельные серверные пакеты, так и компоненты ядра операционной системы, например драйверы сетевых плат и *стек TCP/IP*, отвечающий за выполнение сетевых функций Во FreeBSD стек TCP/IP зарекомендовал себя с наилучшей стороны. В нем реализованы средства защиты, позволяющие *фильтровать пакеты*, в том числе на основании IP-адресов отправителя и получателя (об этом рассказывается в главе 17). Средства фильтрации постоянно совершенствуются в современных операционных системах и жизненно необходимы серверам, часто атакуемым хакерами. Помимо средств защиты самой FreeBSD многие серверные программы обладают собственными механизмами защиты

FreeBSD и неродственные операционные системы

Чтобы решить, какая операционная система является в вашем случае наиболее предпочтительной, следует разбить возможные варианты на две группы: UNIX-подобные системы и остальные. FreeBSD относится к первой группе. Прежде всего нужно определить, имеет ли смысл устанавливать одну из разновидностей UNIX. Если ответ положителен, можно переходить к оценке преимуществ различных систем этой группы.

На сегодняшний день спектр существующих операционных систем чрезвычайно велик. Мы остановимся на наиболее популярных системах, функционирующих на платформах потребительского сегмента рынка (процессоры x86 и PowerPC)

- DOS *Дисковая операционная система* (Disk Operating System DOS) была первой системой такого рода для компьютеров IBM PC и в различных формах "дожила" до сегодняшних дней Все эти формы довольно примитивны в сравнении с современными операционными системами. Их основные достоинства — небольшой занимаемый размер (как на диске, так и в оперативной памяти) и способность выполнять приложения в режиме реального времени, что требует точной синхронизации
- **Windows 9x/Me.** Это семейство Windows-систем больше не разрабатывается. В его основе лежит ядро DOS, в которое интегрированы средства графического интерфейса и вытесняющей многозадачности. Системы Windows 9x/Me достаточно ненадежны и имеют много других недостатков. Тем не менее они все еще широко распространены, в немалой степени из-за доступности популярных пользовательских приложений, работающих только в среде Windows, таких как Microsoft Office.
- **Windows NT/2000/XP** Изначально Windows NT представляла собой полностью переработанную реализацию среды Windows В последующих версиях системы улучшались средства пользовательского интерфейса и повышалась совместимость с популярными программами В 2002 г компания Microsoft позиционировала Windows 2000 как серверную и корпоративную ОС, а Windows XP — как систему для домашнего применения Обе они основаны на Windows NT и гораздо надежнее семейства Windows 9x/Me, основанного на DOS Считается, что FreeBSD надежнее, чем Windows NT/2000/XP, хотя у такого мнения, естественно, немало противников Windows 2000 или XP будет хорошим выбором в том случае, если требуется запускать программы, работающие только в среде Windows
- **OS/2** Несмотря на то что компания IBM не выпускала новых версий OS/2 с 1996 г (Warp 4), эта система остается вполне жизнеспособной По надежности она сопоставима с Windows 2000/XP, вот только способна выполнять лишь старые 16-разрядные приложения Windows, DOS-приложения и программы, написанные специально для OS/2 В среду OS/2 было перенесено множество традиционного программного обеспечения UNIX. В 2001 г лицензию на OS/2 приобрела компания Serenity Systems (`www.serenity-systems.com`) которая выпустила обновленную версию под названием *eComStation*.
- **BeOS** Система BeOS разрабатывалась "с нуля" с учетом последних достижений в области операционных систем, но так и не стала популярной. Круг ее пользователей продолжает сокращаться, хотя у системы остаются верные поклонники BeOS отличается стабильностью и производительностью однако число доступных программ невелико Как и OS/2, BeOS имеет преимущества в той среде, где вычислительная инфраструктура давно налажена В остальных случаях использование системы неоправданно В 2001 г компанию Be, Inc разработчика BeOS, приобрела компания Palm (`www palm.com`).

- Mac OS Mac OS долгое время являлась единственным серьезным конкурентом Windows на рынке настольных систем, но оба семейства операционных систем работают на разном оборудовании в случае Windows это процессоры *x86*, а в случае Mac OS — процессоры 680x0 и (с недавнего времени) PowerPC Mac OS славится своим пользовательским интерфейсом и имеет преданную армию поклонников. Последние версии Mac OS, называемые Mac OS X, основаны на ядре UNIX с немалыми заимствованиями из FreeBSD, так что в определенном смысле MAC OS X можно считать разновидностью UNIX. Отличия, впрочем, остаются довольно существенными, особенно в плане пользовательского интерфейса, поэтому система и была отнесена к рассматриваемой категории Mac OS X может оказаться хорошей альтернативой в том случае, если требуется система с надежным ядром UNIX и возможностью выполнять популярные пользовательские приложения (при условии, что они перенесены в Mac OS X) FreeBSD ориентирована в основном на платформу *x86*, поэтому недоступна для оборудования, на котором традиционно работает Mac OS X (в настоящее время ведутся разработки по переносу FreeBSD на платформу PowerPC)

Сравнивать FreeBSD с перечисленными операционными системами сложно так как они довольно неоднородны Остановимся на наиболее серьезных соперниках — Windows (преимущественно Windows 2000 и XP, хотя Windows 9*x*/Me тоже нужно учитывать из-за огромного количества инсталлированных систем) и Mac OS

В сравнении с Windows достоинства FreeBSD проявляются в основном в серверной части Некоторые из наиболее популярных серверов, в частности почтовый сервер `sendmail` и Web-сервер Apache, ориентированы на платформу UNIX и отлично работают во FreeBSD В последние годы эти программы привлекали меньше внимания со стороны хакеров, чем их Windows-аналоги Например, в 2001 г. Web-серверы Windows подверглись массированному нападению вирусов-червей типа Code Red и Nimda Впрочем, нельзя поручиться, что ситуация не изменится в будущем

Популярность Microsoft Office делает FreeBSD менее предпочтительным выбором чем Windows и Mac OS, когда речь заходит об офисных применениях, например о редактировании текста (по крайней мере, если необходимо обмениваться файлами с другими пользователями Microsoft Office) Альтернативные пакеты наподобие OpenOffice org позволяют читать и записывать файлы в формате Microsoft Office, но могут возникать проблемы с форматированием Если же совместимость с Microsoft Office не требуется, то шансы уравниваются. Как правило, предпочтение отдается той операционной системе, которая лучше поддерживает применяемое программное обеспечение. Очевидными плюсами FreeBSD являются гибкость и сетевые возможности X-среды Но если пользователи имеют опыт работы с Windows или Mac OS, интерфейс FreeBSD покажется им непривычным и не всегда удобным

FreeBSD и другие разновидности UNIX

Если вы окончательно решили отдать предпочтение UNIX, а не Windows, Mac OS или другой системе, то встает следующий вопрос какую разновидность UNIX выбрать? FreeBSD — один из нескольких доступных вариантов, и нужно понимать, чем эта система отличается от других аналогичных систем Во многих случаях любая UNIX-система вполне подходит для решения поставленных задач, но бывают ситуации, когда FreeBSD лучше (или хуже) справляется с конкретной задачей

Как правило, в различных UNIX-системах используется одно и то же программное обеспечение Например, во всех системах работают популярные серверы `sendmail` и Apache, а также X Window System. Ниже перечислено то, что действительно отличает UNIX-системы друг от друга

- **Ядро**. *Ядро* — это программный компонент, составляющий основу операционной системы. Оно служит интерфейсом между пользовательскими программами и аппаратными устройствами компьютера, управляет файлами и процессами и выполняет ряд других низкоуровневых задач. Таким образом, ядро обеспечивает стабильность операционной системы и определяет ее характеристики (например, перечень поддерживаемых аппаратных устройств (некоторые устройства обслуживаются внешними драйверами).
- **Аппаратная поддержка**. UNIX-системы работают не только на платформе *x86*. Производительность процессоров *x86* ставит их на нижнюю ступень, по крайней мере в семействе процессоров универсального назначения. Некоторые UNIX-системы ориентированы на суперкомпьютеры, и обычные операционные системы, такие как FreeBSD, не могут с ними конкурировать. Поддержка конкретных аппаратных устройств (модемов, сетевых и звуковых плат и т.д.) зависит от ядра и вспомогательных программ.
- **Стратегия распространения**. Некоторые разновидности UNIX распространяются как коммерческие продукты, но в основном они представляют собой системы с *открытым исходным кодом*. Это означает, что любой желающий может получить исходный код системы, модифицировать его и распространить дальше. Системы с открытым исходным кодом значительно дешевле в эксплуатации, чем их коммерческие аналоги. Впрочем, в 1990-е годы коммерческие разновидности UNIX упали в цене, а некоторые из них даже стали бесплатными для определенного круга применений.
- **Стандартный набор и конфигурация программного обеспечения**. UNIX существует уже около трех десятилетий, и за это время появилось множество версий популярных утилит, конфигурируемых самыми разными способами. В каждой UNIX-системе свой набор стандартных программ. В одних системах используются конфигурационные сценарии BSD-типа, в других — сценарии System V. Наиболее распространенным почтовым сервером является программа `sendmail`, но в некоторых системах работают другие серверы. Подобные различия определяют уникальность каждой операционной системы семейства UNIX. Если сравнивать UNIX и, скажем, Windows, то эти различия несущественны, но в то же время их более чем достаточно для того, чтобы при переходе из одной UNIX-системы в другую приходилось покупать справочное руководство.

Ядро FreeBSD является производным от открытой реализации исходного ядра AT&T UNIX. Изначально система предназначалась для процессоров семейства Intel *x86*, но впоследствии стала поддерживать родственные семейства процессоров компаний AMD, VIA (Cyrix), Transmeta и других (большей частью не существующих сегодня). Ведутся работы по переносу системы на другие процессоры, например Alpha, но они еще далеки от завершения.

Ядро FreeBSD отличается *монолитностью*, т.е. имеет единую логическую структуру. Вспомогательные компоненты, например драйверы файловой системы, помещаются в так называемое *пространство ядра* и потому могут контролировать аппаратные компоненты и структуры данных ядра. В некоторых разновидностях UNIX применяется иная модель, называемая *микроядром*. В такой схеме ядро имеет меньший размер и взаимодействует с внешними компонентами, реализующими традиционные функции ядра, в частности доступ к файловой системе. У каждого подхода есть свои сторонники.

FreeBSD – это операционная система с открытым исходным кодом. Большинство ее компонентов распространяется на условиях *лицензии BSD* (`http://www.opensource.org/licenses/bsd_license.html`). Эта лицензия разрешает модификацию и последующее распространение исходного кода на тех условиях, которые выберет разработ-

чик. Теоретически это означает, что кто угодно может модифицировать код системы и продать его как коммерческий продукт. Другой популярной лицензией для систем с открытым исходным кодом является *GPL* (GNU General Public License). Она требует, чтобы любые модификации распространялись с соблюдением исходных требований GPL. Такую лицензию имеет ядро Linux, и отсюда проистекают многие различия между FreeBSD и Linux. Во FreeBSD используются программные компоненты, имеющие лицензию BSD, а в Linux больше GPL-компонентов. Это не влияет напрямую на работу операционной системы, но многие утилиты существуют в разных вариантах в зависимости от типа лицензии.

Некоторые разновидности UNIX патентованы. Например, системы Solaris компании Sun и AIX компании IBM являются коммерческими системами. Иногда их можно использовать бесплатно, как в случае Solaris для платформы x86, но чаще всего за лицензию коммерческой версии UNIX приходится платить. Различия между открытыми и коммерческими системами лежат в основном в концептуальной и экономической плоскостях. С функциональной точки зрения FreeBSD во многом эквивалентна своим коммерческим аналогам.

На заметку Коммерческие и открытые системы часто совместимы. Программы, имеющие открытый исходный код, можно запускать в коммерческой операционной системе, и наоборот, коммерческие программы можно запускать в открытой ОС. Тем не менее большинство программ FreeBSD имеет открытую лицензию.

Помимо FreeBSD широко распространены следующие разновидности UNIX:

- **NetBSD**. Это одна из трех операционных систем, производных от 386/BSD. Две другие — FreeBSD и OpenBSD. NetBSD отличается от FreeBSD тем, что перенесена на большее число процессоров. Принципы их лицензирования и общая структура весьма схожи, но FreeBSD остается более популярной, так как активнее рекламируется и чуть проще в инсталляции и применении.
- **OpenBSD**. Эта система славится своей безопасностью. Ее разработчики потратили немало усилий на то, чтобы создать систему, которую не так просто взломать. Как и NetBSD, OpenBSD схожа с FreeBSD по конфигурации, принципам лицензирования и доступному программному обеспечению.
- **Linux**. Linux можно считать отдельным семейством операционных систем с открытым исходным кодом. С технической точки зрения Linux — это лишь ядро; в *дистрибутив* Linux входят ядро и пакеты программ, составляющие операционную систему. По общим принципам конфигурирования и составу программных средств FreeBSD больше отличается от Linux, чем от NetBSD и OpenBSD.

На заметку Пользователи, не знакомые с UNIX, иногда считают FreeBSD дистрибутивом Linux. Это не так. FreeBSD не использует ядро Linux и отличается множеством конфигурационных нюансов. FreeBSD следует считать производной от дистрибутива BSD (Berkeley Software Distribution), тогда как Linux является свободно распространяемой, полностью переработанной реализацией UNIX, гораздо слабее связанной с ранними версиями BSD и AT&T UNIX.

- **Solaris**. Это коммерческая разновидность UNIX, выпущенная компанией Sun и используемая в основном на ее серверных платформах. Серверы Sun, работающие на процессорах SPARC, являются конкурентами высокопроизводительных систем на базе процессоров x86 старших моделей. Операционная система Solaris может работать и на платформе x86, однако набор ее драйверов меньше, чем у FreeBSD. Система Solaris удобна для разработчиков, которые

пишут программы для компьютеров Sun. В остальных случаях более предпочтительным выбором для персональных компьютеров являются FreeBSD и Linux

- **AIX** Это коммерческая разновидность UNIX, выпущенная компанией IBM для своего специализированного оборудования Подобно системам компании Sun, AIX нацелена на верхний сегмент рынка серверов, поэтому мало конкурирует с FreeBSD
- **IRIX** Большинство рабочих станций Silicon Graphics работает под управлением IRIX — коммерческой разновидности UNIX. Ранее основным достоинством этих систем была оптимизированная обработка высококачественной графики, с их помощью создавались спецэффекты для кинофильмов Сегодня в этот сегмент рынка активно вторгаются недорогие компьютеры на базе процессоров *x86*, но позиции Silicon Graphics и IRIX все еще сильны

В целом основными конкурентами FreeBSD являются другие BSD-системы и Linux FreeBSD популярнее своих "родственников" на платформе *x86*, поэтому активнее поддерживается сторонними разработчиками Эта система разрабатывалась с акцентом на простоту инсталляции и администрирования; здесь она опережает NetBSD и OpenBSD Достоинством OpenBSD является улучшенная безопасность, а NetBSD поддерживает не только процессоры *x86* Что касается Linux, то у нее больше отличий в плане аппаратной и программной поддержки и ее лицензирование основывается на несколько иных принципах Для новичков, которым требуется UNIX-система универсального назначения с открытым исходным кодом, FreeBSD и Linux подойдут в одинаковой степени Но Linux развивается гораздо хаотичнее, и различия между ее многочисленными дистрибутивами часто сбивают с толку неопытных пользователей

FreeBSD вряд ли можно считать конкурентом коммерческих разновидностей UNIX ориентированных на компьютеры, которые превосходят по производительности системы на базе процессоров *x86*. Если в организации используются высокопроизводительные серверы Sun, IBM или SGI в сочетании с недорогими персональными компьютерами, то на последних можно установить FreeBSD, поскольку процедуры администрирования FreeBSD и коммерческих разновидностей UNIX во многом схожи Иногда системы *x86* старших моделей напрямую конкурируют с системами Sun, IBM и SGI младших моделей. Достоинством первых является низкая цена, но для них сложнее заключать контракты на сервисное обслуживание

Программная среда

В предыдущих разделах рассказывалось о том какое место занимает FreeBSD в ряду других операционных систем Это позволит читателям определиться с выбором Если в итоге предпочтение отдано FreeBSD, то нужно выяснить, как она взаимодействует с другими операционными системами. Такое взаимодействие может происходить как между компьютерами сети, так и в пределах одного компьютера. Полезно также узнать кое-что об истории FreeBSD чтобы понять главные принципы ее разработки, ее слабости и недостатки

История разработки FreeBSD

История FreeBSD, как и UNIX вообще, довольно запутанна Она начинается в 1969 г с появления первой (довольно примитивной) версии UNIX, разработанной компанией AT&T Далее свой вклад в систему вносили энтузиасты Калифорнийского университета в Беркли и многочисленные добровольцы, существующий код неодно

кратно менялся, пополнялся набор системных утилит Исходная версия AT&T UNIX породила несколько вариантов (или *ветвей*, как их еще называют) Хорошие идеи, появлявшиеся в одной ветви, часто воплощались в других ветвях В истории UNIX были периоды, когда поставщики систем работали над устранением различий между ветвями Тем не менее сегодня UNIX — это не одна операционная система, а целое семейство

На заметку

С технической точки зрения название *UNIX* относится к официальным версиям именно этой системы. Иногда официальные версии обозначают словом *UNIX* (все прописные), а другие однотипные системы называют *Unix* (с заглавной буквы). Но такое различие почти незаметно поэтому в данной книге словом *UNIX* обозначаются официальные версии либо все семейство в целом, а когда нужно подчеркнуть что речь идет также о других подобных операционных системах, например Linux, употребляется выражение "UNIX подобные системы"

Итак, в 1969 г Кен Томпсон (Ken Thompson) и Деннис Ритчи (Dennis Ritchie) написали первую версию UNIX. Очередной крупной вехой на пути к появлению FreeBSD стала установка четвертой версии UNIX в Калифорнийском университете в Беркли в 1974 году В течение следующих нескольких лет преподаватели, научные сотрудники и студенты университета знакомились с операционной системой, писали программы для нее и даже модифицировали саму систему В 1977 г все наработки были включены в единый пакет программ названный *BSD (Berkeley Software Distribution)*

В 1978 г был выпущен пакет *2BSD*, содержащий улучшенный и расширенный набор утилит исходного дистрибутива Последующие разработки привели к появлению пакетов 2 11BSD, 3BSD, 4BSD. На очереди было название 5BSD, однако компания AT&T возражала против этого, т.к боялась потенциальной путаницы с *System V* своим вариантом UNIX В результате следующие версии стали называться 4.1BSD 4.2BSD и т.д. В это время и появился *TCP/IP* — базовый стек сетевых протоколов лежащих в основе современной сети Internet

Упомянутые BSD-пакеты представляли собой дополнения к ОС UNIX компании AT&T. Это не были полноценные операционные системы Кроме того, лицензионные требования постоянно ужесточались Один из вариантов пакета 4.3BSD, известный как 4 3BSD-Tahoe, был доступен только при условии приобретения исходной лицензии AT&T, которая определяла правила распространения ОС UNIX в семидесятые и восьмидесятые годы. Но в 1989 г ситуация изменилась разработчики из Беркли выпустили пакет Networking Release 1 который допускал свободное распространение кода независимо от того имел ли получатель лицензию на исходный код AT&T Это был первый и очень важный шаг на пути к современным системам с открытым исходным кодом Без изменений в правилах лицензирования, впервые воплощенных в Networking Release 1, не было бы современной FreeBSD

В 1991 г появился пакет Networking Release 2 Несмотря на сходство названий, изменения оказались разительными Это была практически полноценная операционная система; не хватало лишь шести файлов с исходными кодами ядра AT&T UNIX В течение шести месяцев программист Билл Джолиц (Bill Jolitz) переписал их, назвав полученный пакет *386/BSD*. Эта система является непосредственной предшественницей NetBSD и FreeBSD (OpenBSD отделилась от ветви NetBSD в середине 90-х) Она предназначалась для работы на процессорах Intel x86, точнее, 80386 — самой последней на то время модели Как уже отмечалось выше, FreeBSD по-прежнему тесно связана с платформой x86, хотя и предпринимаются попытки ее переноса на другие процессоры

С 1993 г (дата рождения системы как отдельного проекта) FreeBSD целенаправленно развивается независимо от других систем Она, конечно же заимствует что-то

из других проектов с открытым исходным кодом; иногда заимствуют ее код Старший номер версии увеличивается в среднем каждые несколько лет. В начале 2002 г была выпущена версия 4.5, а версия 5 0 появилась в конце того же года Именно эти две версии и рассматриваются в нашей книге

Взаимодействие с другими компьютерами по сети

Одной из вех в истории (или, если хотите, предыстории) FreeBSD было появление стека протоколов TCP/IP в 4.2BSD Он послужил основой для аналогичных стеков других операционных систем Даже некоторые сетевые утилиты компании Microsoft до сих пор отображают сообщения об авторских правах, свидетельствующие о том, что эти утилиты заимствованы из BSD. С учетом этих обстоятельств, казалось бы, у FreeBSD не должно быть никаких проблем при взаимодействии с другими системами по сети В некоторых случаях это действительно так, но есть нюансы и ограничения которые обязательно следует учитывать

- **Альтернативные стеки сетевых протоколов**. TCP/IP — не единственный существующий стек сетевых протоколов. В некоторых локальных сетях используются другие семейства протоколов, например AppleTalk компании Apple, NetBEUI в Windows и IPX компании Novell Поддержка этих стеков во FreeBSD ограничена в сравнении с TCP/IP К счастью, немногие сетевые утилиты сегодня работают исключительно с альтернативными протоколами К примеру, в Windows утилиты совместного доступа к файлам и принтерам поддерживают как NetBIOS, так и TCP/IP. Этот факт учитывает программа Samba (описывается в главе 18) предоставляющая Windows-клиентам доступ к файлам.
- **Непостоянство стандартов**. Редко какой поставщик использует сторонний код в том виде, в каком он был получен Даже авторы время от времени переделывают свои творения По этой причине исходный стек TCP/IP, появившийся в 4 2BSD, за годы эксплуатации "мутировал", и в каждой следующей операционной системе использовалась несколько иная его реализация, иногда даже совершенно новая Но при всем многообразии изменений нельзя не отметить тот факт, что большинство операционных систем может взаимодействовать друг с другом по протоколам TCP/IP С практической точки зрения важно то что FreeBSD как одна из основных серверных систем крайне редко бывает несовместима с другими системами, с которыми ведется работа по сети. Проблемы чаще возникают с конкретными протоколами, применяемыми клиентом или сервером Например, Web-страницы, разработанные для броузера Internet Explorer компании Microsoft, не всегда корректно отображаются в броузерах доступных во FreeBSD, таких как Mozilla или Konqueror.
- **Проблемы с безопасностью**. Протоколы семейства TCP/IP разрабатывались во времена взаимного доверия, поэтому в них заложено очень мало средств защиты В результате с наступлением эры Internet в сетевом программном обеспечении стали постоянно обнаруживаться "дыры", позволяющие злоумышленникам получать несанкционированный доступ к компьютерам В плане безопасности у FreeBSD хорошая репутация, но это не значит, что вопросы безопасности можно игнорировать Неправильная конфигурация системы почти наверняка приводит к появлению "дыр". Те или иные аспекты защиты системы рассматриваются в разных главах книги, но особенно детально эта тема освещается в главах 17 и 29.

В целом FreeBSD прекрасно подходит для выполнения многих сетевых функций В ней реализован стабильный и эффективный стек TCP/IP, что особенно важно для серверов. Написано множество серверных программ, позволяющих пользователям

FreeBSD взаимодействовать с другими системами Что касается безопасности то FreeBSD обладает хорошими средствами защиты

Внимание

Безопасность — весьма относительное понятие Заявления "FreeBSD — защищенная система" нельзя воспринимать как стопроцентную гарантию защиты. В лучшем случае можно надеяться на то, что система капитулирует лишь перед наиболее опытными и настойчивыми хакерами. Кроме того, стандарты безопасности постоянно меняются по мере выявления все новых ошибок в программах наряду с изобретением новых способов взлома Система, которая сегодня кажется разумно защищенной, завтра может оказаться абсолютно беззащитной перед лицом очередной угрозы

Сосуществование с другими операционными системами на одном компьютере

С самого начала разработки BSD-систем возникла потребность в сосуществовании UNIX с другими операционными системами на одном компьютере. Например, на PDP 11/45, первом UNIX-компьютере в Беркли, ОС UNIX выполнялась лишь треть рабочего времени. Остальное время отводилось другой операционной системе (RSTS) Но для компьютера, работающего в режиме разделения времени, это чрезвычайно неуклюжий подход, поэтому на большинстве UNIX компьютеров в 1970–1980 гг. выполнялась одна операционная система С появлением 386/BSD вопрос сосуществования различных операционных систем стал гораздо острее. Многие программисты любители имели всего один компьютер и часто сталкивались с необходимостью писать программы для более популярной DOS (позднее — Windows) Такое положение дел сохранилось и сегодня. Если FreeBSD устанавливается на выделенный компьютер (сервер или специализированную рабочую станцию), то задача упрощается, так как использовать другие операционные системы не предполагается. Если же планируется время от времени запускать программы в другой ОС, то нужно разобраться как FreeBSD работает с жесткими дисками и как на персональных компьютерах загружается операционная система

Одна из ключевых концепций мультисистемных компьютеров — *разделы* Если представить жесткий диск как шкаф для бумаг, то раздел – это один из его ящиков, часть большого документохранилища, содержащая взаимосвязанные файлы. Разделы создаются при подготовке диска к использованию Это называется разбивкой на разделы После того как раздел создан, его не так то легко модифицировать. Следовательно, важно продумать такую структуру жесткого диска, которая в обозримом будущем отвечала бы потребностям пользователя

На заметку

Есть утилиты, позволяющие менять размеры существующих разделов Это, в частности, коммерческие программы Partition Magic (`www.powerquest com`) и Partition Commander (`www.v-com.com`) Среди программ с открытым исходным кодом назовем FIPS (`http://www.igd.fhg.de/~aschaefe/fips/`) и GNU Parted (`http://www.gnu.org/software/parted`) Это более простые программы с ограниченными возможностями, но свою работу они выполняют. Особенно полезны они в том случае, когда FreeBSD устанавливается в системе, где уже инсталлирована DOS Windows OS/2 Linux или другая операционная система.

За много лет были придуманы различные схемы разбивки на разделы В основном они разрабатывались независимо друг от друга для конкретных аппаратных платформ

На персональных компьютерах поддерживается три типа разделов. Их существование не так уж необходимо, просто предыдущая схема была расширена, когда стало очевидно что ее возможности исчерпаны

- **Первичный раздел** Исходный тип раздела персональных компьютеров теперь называется *первичным* разделом Всего таких разделов может быть четыре Некоторые операционные системы, такие как DOS и Windows, должны загружаться с первичного раздела, расположенного на первом физическом диске. FreeBSD тоже должна находиться в первичном разделе но он может быть не первым. Есть операционные системы, например Linux и OS/2, которые могут загружаться с расширенного раздела Все эти нюансы нужно учитывать планируя инсталляцию системы.
- **Расширенный раздел** В определенном смысле *расширенный* раздел представляет собой всего лишь особый тип первичного раздела Он занимает одну из четырех доступных областей первичных разделов и служит для последующей разбивки на логические разделы.
- **Логический раздел**. *Логические* разделы создаются внутри расширенных. Запись для расширенного раздела в таблице разделов необходима для того, чтобы зарезервировать место под один или несколько логических разделов Их число ограничивается емкостью жесткого диска и способом адресации разделов, применяемым в операционной системе. Например, в DOS и Windows разделы помечаются буквами (именами дисков), поэтому теоретически их может быть 26 (число букв английского алфавита) В UNIX-подобных системах число логических разделов зависит от ядра и содержимого каталога /dev. В любом случае редко какая система даже приближается к теоретическому пределу. Поскольку логические разделы находятся в расширенном разделе, они должны последовательно занимать смежные области жесткого диска

В DOS и Windows жесткий диск часто представляет собой один большой первичный раздел Но если на одном компьютере требуется установить FreeBSD и какую-то другую операционную систему, то придется создавать дополнительные разделы В частности, под FreeBSD должен быть отведен один из первичных разделов Одна из возможных конфигураций изображена на рис 1 1 В данном случае FreeBSD и Windows занимают по одному первичному разделу, а в расширенном разделе Windows созданы два логических диска.

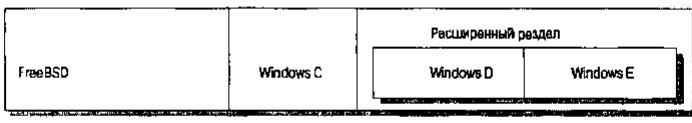


Рис 1 1 Разбивка на разделы позволяет отделять операционные системы друг от друга и создавать области хранения для различных типов данных

На заметку Компьютеры на базе новых 64-разрядных процессоров Intel IA64 поддерживают дополнительную, более гибкую схему разбивки на разделы

В самой FreeBSD ситуация намного сложнее Прежде всего нужно понять, что в UNIX-системах разделы часто используются для обособленного хранения различных типов данных Это позволяет монтировать каталоги с различными параметрами и защищать данные в случае переполнения или повреждения других разделов Например, каталог /home, хранящий файлы пользователей часто отделяют от корневого каталога

(/). Во FreeBSD как и в других разновидностях UNIX каждый раздел *монтируется* к определенному каталогу. Нет логических дисков, как в DOS или Windows. По этой причине FreeBSD-системы часто требуют нескольких разделов, даже если на компьютере установлена только FreeBSD.

Традиционно во FreeBSD применяется схема разбивки на разделы, отличающаяся от стандартной схемы для персональных компьютеров. Обычно разбивка осуществляется в пределах одного первичного раздела. Его подразделы аналогичны логическим дискам расширенного раздела. Работать с подразделами может только BSD-система. В главе 2 будут описаны типы подразделов, которые можно создавать во FreeBSD.

Если нужно обмениваться данными с другими операционными системами, лучше выделить для этого отдельный раздел. К сожалению, в разных операционных системах для хранения данных применяются разные *файловые системы*. Файловая система, занимающая целый раздел, ведет учет файлов, следит за их расположением и управляет служебной информацией, такой как дата создания файла. Как правило, для перекрестного обмена данными подходит файловая система *FAT (File Allocation Table)* из DOS, поскольку ее поддерживают все распространенные операционные системы персональных компьютеров. Лишь немногие системы, не относящиеся к семейству UNIX, поддерживают *FFS (Fast File System)*, собственную файловую систему FreeBSD. Вообще говоря, в разных версиях UNIX применяется несколько иная реализация FFS, так что, даже если планируется установить на одном компьютере вместе с FreeBSD еще одну разновидность UNIX, может потребоваться использовать FAT в качестве общей среды обмена файлами.

Аппаратная среда

Прежде чем инсталлировать FreeBSD, нужно узнать аппаратные требования этой системы. Платформа *x86* чрезвычайно разнообразна, поэтому в компьютерах часто присутствуют компоненты, не поддерживаемые во FreeBSD. Как правило, это не мешает инсталлировать систему, но ее возможности будут ограничены.

Если в процессе инсталляции системы обнаружено неподдерживаемое устройство, замените его, прежде чем продолжать инсталляцию. Это позволит избежать последующего переконфигурирования системы.

Типы и производительность процессоров

"Сердцем" компьютера является центральный процессор. Как уже говорилось выше, FreeBSD изначально предназначалась для работы на процессорах Intel *x86*, по крайней мере на 32-разрядных процессорах, начиная с 80386. В 2002 г. FreeBSD поддерживала процессоры Intel 80386, 80486, Pentium, Pentium Pro, Pentium II, Pentium III, Pentium 4 и серию Celeron со всеми разновидностями. Помимо этого FreeBSD работает на аналогичных процессорах других компаний, таких как AMD, VIA, Transmeta, Cyrix, IDT и NexGen (последние три обанкротились, но их разработки внедрены в процессоры AMD и VIA). Некоторые компании, например Evergreen, продают комбинированные системы и позволяют "разгонять" старые материнские платы до ранее недоступных скоростей. FreeBSD работает с такими процессорами, но могут возникать трудности с выбором материнских плат.

Существуют проекты по переносу FreeBSD на другие платформы (подробнее об этом можно узнать по адресу `http //www.freebsd.org/platforms/`), в том числе Intel IA64, AMD *x86*-64 (ответ компании AMD на архитектуру IA64), Alpha компании Compaq, PowerPC (PPC) компании Motorola и UltraSparc компании Sun. В настоящее

время архитектура x86 остается наиболее популярной но появились признаки того что со временем она будет вытеснена 64-разрядными платформами. Компания Intel продвигает процессоры платформы IA64 под названием *Itanium*. Компания AMD готовит конкурирующую платформу x86 64 (*K8*, или *Hammer*) Эти процессоры, без со мнения, будут играть очень большую роль в недалеком будущем К счастью, функ ционирование FreeBSD не зависит от используемой платформы, хотя тип процессора нужно учитывать при выборе скомпилированных программ. При инсталляции FreeBSD на альтернативных платформах может меняться порядок разбивки на разделы и установки системы. Впрочем, в этой книге мы сосредоточимся на платформе x86 как наиболее популярной.

Что касается скорости работы процессора, то для FreeBSD здесь нет ограничений Система работает даже на самых низкоскоростных процессорах 80386, хотя такое применение в наши дни мало кого заинтересует. Систему на базе процессора 80386 можно использовать разве что в качестве выделенного брандмауэра для домашней сети или DHCP-сервера небольшой сети Для обработки графики и выполнения современных ресурсоемких программ производительности такого процессора недостаточно. Подбирая комплектацию нового компьютера, старайтесь устанавливать самые новые процессоры компаний Intel, AMD, VIA или Transmeta. Если планируется запускать программы, предъявляющие повышенные требования к производительности, выби райте более новые модели процессоров.

Совет

Цена на процессоры резко снижается при переходе от самой последней модели к предыдущей и гораздо меньше — при последующем сниже нии скорости процессора. Таким образом, оптимальным выбором с точки зрения соотношения цена/производительность часто являются процессоры промежуточного звена Планка "промежуточного" уровня повышается чуть ли не ежемесячно, так что конкретные рекомендации по поводу выбора оптимального процессора давать сложно

Требования к оперативной памяти

Как и в отношении центрального процессора, требования к оперативной памяти, предъявляемые во FreeBSD, умеренны по современным меркам В официальной документации говорится, что для инсталляции системы требуется минимум 5 Мбайт ОЗУ, а для работы самой системы требуется минимум 4 Мбайт Сегодня это кажется смехотворным Такие объемы памяти достаточны лишь для самых тривиальных приложений Современные компьютеры оснащаются минимум 128 Мбайт ОЗУ Этого более чем достаточно для инсталляции FreeBSD, загрузки графической среды и запуска большинства серверов (хватит даже 64 Мбайт). Но в определенных ситуациях требуется дополнительная память

- **Многопользовательская среда**. Если в системе одновременно работает множество пользователей (например, по сети), то для обслуживания пользовательских приложений и графических оболочек нужна дополнительная память.
- **Крупные серверы** Для большинства серверов достаточно 64 Мбайт ОЗУ но требования к памяти повышаются по мере роста вычислительной нагрузки Если на одном компьютере функционируют файловый сервер, Web-сервер, почтовый сервер и все они обслуживают сотни или тысячи пользователей, то минимально допустимый объем оперативной памяти будет гораздо больше, чем 128 Мбайт. Конкретный показатель зависит от предполагаемого уровня загруженности
- **Ресурсоемкие программы** Некоторые программы, например анимационные или моделирующие пакеты, требуют больших объемов памяти даже если с ними

работает один пользователь. Конкретные значения указаны в документации к программе.

Как правило, с повышением требований к памяти возрастают и требования к скорости центрального процессора, но так бывает не всегда.

Жесткие диски

При инсталляции операционной системы нужно учитывать объем дискового пространства. Выше уже описывались принципы разбивки на разделы, применяемые во FreeBSD. Подробнее об этом будет рассказываться в главе 2. Но прежде чем приступать к инсталляции, следует выбрать тип жесткого диска и оценить общий объем дискового пространства, отводимый под операционную систему.

Диски SCSI и EIDE

На многих современных персональных компьютерах установлены жесткие диски *EIDE (Enhanced Integrated Drive Electronics)*. Материнские платы обязательно содержат контроллеры EIDE, поэтому подключение дисков не представляет проблемы. Но существует и альтернативный тип жестких дисков *SCSI (Small Computer System Interface)*. Очень немногие материнские платы оснащены соответствующими разъемными соединениями, поэтому для работы со SCSI-дисками нужно установить *SCSI-адаптер* (обычно это PCI-плата). FreeBSD поддерживает оба интерфейса — SCSI и EIDE. У каждого из них свои достоинства и недостатки.

Основное преимущество стандарта EIDE — стоимость. Поскольку соответствующие контроллеры встроены в материнские платы, их не приходится покупать, как в случае SCSI (контроллер можно добавить для повышения производительности или при наличии в системе более четырех жестких дисков). Жесткие диски EIDE тоже недороги в сравнении с аналогичными SCSI-дисками. Общая экономия составляет от 100 до 500 долларов, иногда даже больше. Все контроллеры EIDE поддерживают единый минимальный стандарт низкоуровневого форматирования жестких дисков, поэтому диски работают на любых контроллерах по крайней мере в минимальной низкоуровневой конфигурации. Но чтобы обеспечить наилучшую производительность, следует использовать жесткий диск, рассчитанный на имеющийся контроллер.

Основным недостатком EIDE является то, что к одному *шлейфу* (кабель, подключающийся с другой стороны к материнской плате или контроллеру EIDE) можно подключить лишь два EIDE-устройства. В большинстве материнских плат всего два шлейфа, что обеспечивает подключение четырех EIDE-устройств. Это вызывает очень большие неудобства, если в системе есть дисководы CD-ROM, съемные диски, ленточные накопители и т.д. Обойти ограничение можно, добавив еще один контроллер, но ему требуются отдельный PCI-слот и линия запроса прерывания, а это тоже ограниченные ресурсы. Кроме того, стандарт EIDE разрешает одновременный доступ лишь к одному устройству на шлейфе, поэтому в системах с большим числом пользователей EIDE-устройства становятся узким местом.

На заметку

Раньше контроллеры EIDE поддерживали лишь режимы доступа *PIO (Programmed Input/Output)*, которые требуют участия процессора в операциях обмена данными. Это делало стандарт EIDE малопригодным в случае многозадачной операционной системы, такой как FreeBSD. Современные контроллеры поддерживают более оптимальные режимы *DMA (Direct Memory Access)*, при которых контроллер может записывать данные непосредственно в память. Эта поддержка должна также обеспечиваться драйвером EIDE.

SCSI — гораздо более гибкий интерфейс в сравнении с EIDE. Он допускает подключение от 7 до 15 устройств к одному шлейфу, в зависимости от разновидности интерфейса, что делает его гораздо более приемлемым вариантом для компьютера с большим числом физических дисков. SCSI поддерживает одновременный доступ к нескольким устройствам одного шлейфа, а жесткие диски SCSI обычно работают быстрее, чем диски EIDE аналогичной емкости. Для SCSI предлагается более широкий диапазон устройств. Например, существуют сканеры SCSI, а ленточные накопители моделей верхнего уровня почти всегда имеют интерфейс SCSI. Большинство SCSI-адаптеров всегда работает в режиме DMA.

Недостатком SCSI является стоимость. SCSI-адаптер дорог сам по себе (от 50 до 250 долларов), к тому же SCSI-устройства обычно дороже своих EIDE-аналогов. В интерфейсе SCSI не предусмотрен минимальный стандарт низкоуровневого форматирования, поэтому FreeBSD должна поддерживать тот SCSI-адаптер, который установлен в системе.

В целом стоимость EIDE-дисков позволяет применять их в системах нижнего и среднего уровня. Если в компьютере установлен EIDE-контроллер, для которого FreeBSD поддерживает режим DMA, то однодисковая система обеспечит хорошую производительность. В случае крупного сервера, когда требуется улучшенная производительность работы с диском, повышенная стоимость SCSI может оказаться оправданной.

На заметку На одном компьютере разрешается использовать оба интерфейса. Например, к SCSI-контроллеру могут быть подключены высокопроизводительные жесткие диски и ленточный накопитель, а к EIDE контроллеру — недорогие дисководы CD-ROM и Zip.

В будущем важную роль начнут играть другие дисковые интерфейсы. В частности, интерфейс IEEE-1394 (FireWire) может заменить SCSI, по крайней мере в некоторых случаях. Жесткие диски FireWire доступны уже сегодня, однако поддержка этого интерфейса во FreeBSD 5.0 весьма ограничена.

Чтобы выяснить, какие EIDE-контроллеры и SCSI-адаптеры поддерживаются, просмотрите файл `HARDWARE.TXT`, входящий в комплект поставки FreeBSD. Учтите, что в нем указываются микропроцессорные наборы, а информация о конкретных моделях устройств может быть не приведена. Посмотрите названия микросхем на PCI-плате, чтобы узнать, с чем вы имеете дело. Если на компьютере уже инсталлирована Windows, выберите элемент Система (System) в панели управления. Затем щелкните на вкладке Устройства (Device Manager) и раскройте список Контроллеры жестких дисков (Hard disk controllers). Как видно из рис. 1.2, в компьютере установлен EIDE-контроллер VIA (*IDE* — это старый стандарт, трансформировавшийся в EIDE, но в некоторых утилитах используется старая терминология).

На заметку Нет необходимости искать драйверы для конкретных устройств EIDE или SCSI. К примеру, если поменять один EIDE-дисковод CD-ROM на другой, тоже EIDE, то обновлять драйверы FreeBSD не понадобится, поскольку оба дисковода используют стандартизированный набор команд. Драйверы нужны для конкретных *типов* устройств, таких как жесткий диск или дисковод CD-ROM. Эти драйверы включены в стандартное ядро FreeBSD, поэтому применяются практически незаметно для пользователя.

Общие требования к объему дискового пространства

Важно выяснить возможности EIDE-контроллера или SCSI-адаптера системы, чтобы избежать неприятных проблем с дисками. Но независимо от типа интерфейса остается вопрос, связанный с общим объемом дискового пространства. Официально для FreeBSD требуется 100 Мбайт дисковой памяти. Но это минимально необходимый объем. Возможности такой системы крайне ограничены. Обычно требуется не менее 1 Гбайт. Если же инсталлируется много программного обеспечения, то даже 2

или 3 Гбайт не будут излишними Дополнительное пространство может понадобиться для хранения пользовательских или серверных данных. Потребности в дисковой па мяти определяются функциями которые выполняет система

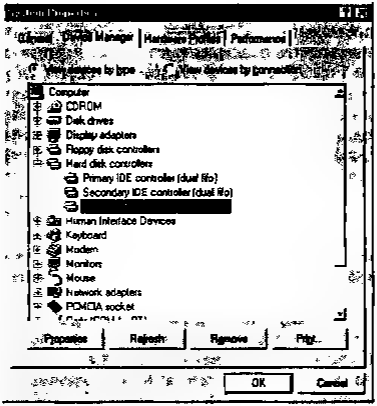


Рис 1 2. В Windows можно быстро узнать тип имеющегося оборудования

Современные жесткие диски имеют емкость не менее 20 Гбайт, поэтому при ин сталляции FreeBSD никаких трудностей не возникает. Если же FreeBSD инсталлиру ется в дополнение к уже установленной системе, то есть два варианта.

- **Повторная разбивка диска на разделы.** Можно воспользоваться специальными утилитами, например коммерческой программой PartitionMagic или свободно распространяемой программой FIPS, для сокращения размеров существующего раздела и выделения пространства под FreeBSD. Это эффективный и экономный подход если на диске достаточно свободного пространства но одновременно и не лишенный риска Утилиты динамической разбивки на разделы иногда сбоят вследствие чего происходит потеря данных. Прежде чем запускать такую утилиту, создайте резервную копию существующих данных.
- **Добавление нового диска.** Можно добавить новый жесткий диск, который будет использоваться либо исключительно под FreeBSD, либо совместно с текущей операционной системой Такой подход безопаснее, чем изменение существующих разделов и добавление новых на одном диске, но на практике он может оказаться неосуществимым из за того, что уже достигнут предел возможного числа дисковых устройств

В любом случае при инсталляции FreeBSD нужно установить *системный загрузчик*. Это очень простая программа, загружающая одну или несколько операционных систем Загрузчик FreeBSD позволяет выбрать систему, запускаемую по умолчанию при включении питания. Эту программу можно применять в сочетании со сторонними загрузчиками, такими как BootMagic компании PowerQuest (`http://www.powerquest com/partitionmagic/`), System Commander компании V-Com (`http://www.v-com com/product/sc7 ind.html`) или LILO операционной системы Linux Подробнее об установке и конфигурировании загрузчика FreeBSD говорится в главе 4

Графическое оборудование

Чтобы на экране появилось изображение, FreeBSD должна поддерживать имеющуюся плату видеоадаптера. Что касается монитора, то он очень редко вызывает проблемы и не требует наличия специальных драйверов (тем не менее, как будет рассказываться в главе 13, определенные его установки нужно настраивать).

Все современные видеоплаты поддерживают определенные базовые режимы, в которых может работать и FreeBSD. Это, в частности, текстовый режим и режим с низким разрешением *VGA* (640×480). Иногда поддерживаются стандартные графические режимы с более высоким разрешением. Текстовый режим можно применять, например, на сервере, но пользователи рабочих станций предпочитают графическую среду. VGA-режимы редко удовлетворяют современных пользователей, поэтому для серьезной системы необходима более мощная графическая плата, поддерживаемая FreeBSD (точнее, X-средой).

Графические средства, хоть и включены в стандартный комплект FreeBSD, в действительности являются частью независимого проекта *XFree86* (`www.xfree86.org`). Пакет XFree86 используется и другими UNIX-системами с открытым исходным кодом, в частности NetBSD, OpenBSD и Linux. Он также доступен в коммерческих разновидностях UNIX (в них обычно имеются еще и собственные X-серверы) и даже в системах других типов, включая Windows, Mac OS и OS/2.

Во FreeBSD подсистема XFree86 взаимодействует большей частью непосредственно с аппаратными устройствами, поэтому она должна поддерживать имеющуюся видеоплату. Если средства поддержки реализованы не в полном объеме, производительность графической подсистемы существенно снизится. Это может привести к тому, что FreeBSD обеспечит меньшее разрешение, чем в Windows, или меньшую частоту обновления экрана. Все современные графические микропроцессорные наборы располагают средствами ускорения основных графических операций, таких как перемещение объектов по экрану. Если же их драйверы несовершенны, эти средства могут не поддерживаться, вследствие чего пострадает производительность графической подсистемы. Подробную информацию о микропроцессорных наборах, поддерживаемых пакетом XFree86, можно найти по адресу `http://www.xfree86.org/current/Status.html`.

На этой Web-странице приведена информация о самых новых драйверах. В зависимости от версии FreeBSD в систему может быть включен старый набор драйверов XFree86, поэтому иногда оказывается, что на Web-странице говорится о существовании драйвера, хотя в конкретной системе его нет. В этом случае нужно обновить пакет XFree86 или хотя бы драйвер видеоплаты.

Как и в случае устройств EIDE и SCSI, список поддерживаемых видеоплат формируется на основе микропроцессорных наборов, хотя изготовители видеоплат не всегда публикуют эту информацию. Следовательно, чтобы собрать необходимую информацию, нужно посмотреть надписи на микросхемах видеоплаты или, если на компьютере установлена Windows, просмотреть информацию о видеоадаптере в панели управления. Некоторые изготовители популярных видеоплат выпускают собственные наборы микросхем, поэтому найти драйверы для них несложно.

Приобретая видеоплату, остерегайтесь самых последних моделей. Разработчики XFree86 часто отстают на несколько месяцев, поэтому последние модели могут поддерживаться плохо или вообще не поддерживаться.

В процессе инсталляции система пытается обнаружить и сконфигурировать видео плату, как описывается в главе 2. Если эта процедура выполняется с ошибками или возникает необходимость изменить конфигурацию платы, отредактируйте файлы конфигурации XFree86 (об этом рассказывается в главе 13).

Сетевое оборудование

Многие компьютеры, работающие под управлением FreeBSD, играют роль серверов или по крайней мере сетевых рабочих станций. В настоящее время наиболее распространенным сетевым стандартом является Ethernet. Существует много версий этого стандарта, отличающихся двумя ключевыми особенностями:

- **Скорость передачи данных.** Самая низкая скорость в сетях Ethernet — 10 Мбит/с. Самая высокая из доступных на сегодняшний день — 1000 Мбит/с (1 Гбит/с). Сейчас наиболее распространены стомегабитные сети, хотя гигабитные технологии завоевывают все большую популярность.
- **Среда передачи данных.** В сетях Ethernet данные передаются по кабелям различных видов. Раньше использовались две разновидности коаксиального кабеля: *толстый* и *тонкий*. Оба напоминают провода, используемые в кабельном телевидении, но отличаются по толщине. Такие сети имеют *шинную топологию*, в которой сетевой кабель последовательно соединяет устройства друг с другом. Коаксиальные кабеля сейчас используются редко. Их вытеснила *витая пара*, напоминающая телефонный провод, но с более широкими разъемными соединениями. В таких сетях имеется центральное устройство, называемое *концентратором* или *коммутатором*, которое соединяет все остальные устройства по топологии "звезда" (рис. 1.3). В последнее время все активнее используются оптоволоконные кабели в качестве альтернативы витой паре в гигабитных сетях Ethernet. Такие сети, как правило, тоже имеют звездообразную топологию.

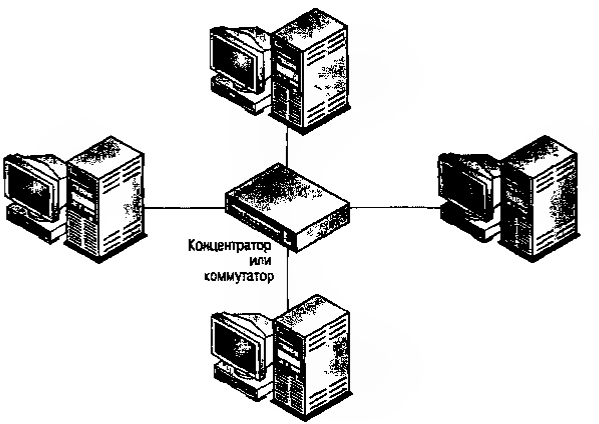


Рис. 1.3. Топология "звезда" упрощает выявление проблем с сетевым кабелем, поскольку сбои локализуется в одном звене

Большинство современных сетей создается на основе стомегабитной витой пары, хотя иногда применяются старые десятимегабитные кабели (витая пара или коаксиал). Если нужна более высокая скорость работы, перейдите на гигабитные технологии.

К счастью, для FreeBSD не важны ни тип кабелей, ни топология. Нужно лишь убедиться в том, что сетевая плата Ethernet поддерживается системой (в некоторых материнских платах верхнего уровня имеется встроенная поддержка Ethernet). В уже упоминавшемся выше файле `HARDWARE.TXT` указано, какие Ethernet-платы поддерживаются во FreeBSD. Как и в случае жестких дисков и видеоадаптеров, в файле описывается поддержка микропроцессорных наборов, установленных на плате, поэтому придется изучать надписи на микросхемах или, если есть возможность, просмотреть информацию в панели управления Windows.

На заметку

FreeBSD должна поддерживать лишь сетевую плату, тогда как поддержка коммутатора или концентратора не требуется. Что касается серверного сетевого оборудования, то FreeBSD должна поддерживать конкретный тип сервера, например файловый сервер или сервер печати.

В последнее время широкую популярность приобретают беспроводные устройства, особенно в домашних и небольших офисных системах. Многие из этих устройств поддерживаются во FreeBSD, и с логической точки зрения они функционируют примерно так же, как и обычные платы Ethernet. Информация о доступных беспроводных устройствах содержится в том же файле `HARDWARE.TXT`.

Многие сетевые установки можно задать на этапе инсталляции системы, как описывается в главе 2. Последующая модификация этих параметров рассматривается в главе 14.

Еще один тип сетевых устройств заслуживает особого внимания. *модемы*. Модемы служат средством подключения компьютеров к сети Internet через маршрутизаторы провайдеров. FreeBSD поддерживает все внешние модемы RS-232 и некоторые внутренние модемы. Впрочем, многие внутренние модемы часто являются *программными*. Для этих устройств нужен специальный драйвер, который выполняет функции, в обычных модемах реализованные на аппаратном уровне. Такие драйверы — редкость для FreeBSD. Сколько бы ни стоили такие модемы, их лучше избегать.

На заметку

Модемы, предназначенные для широкополосного доступа в Internet (кабельные и DSL-модемы), часто имеют Ethernet-интерфейсы. Для FreeBSD эти устройства выглядят как часть обычной сети Ethernet и потому не требуют специальных драйверов. Но у этого правила есть два исключения. Первое, если модем является внутренним или имеет интерфейс USB, для него нужен отдельный драйвер, которых практически нет для FreeBSD. Второе, некоторые провайдеры широкополосного доступа применяют протокол PPPoE (PPP over Ethernet) для назначения IP-адресов. Сам по себе такой протокол не нужен широкополосному модему, но он поддерживается во FreeBSD.

Принтеры

До сих пор рассматривались внутренние устройства, т.е. устанавливаемые внутри компьютера. Существуют, естественно, и внешние устройства. Среди них наиболее широко распространены принтеры. Аппаратная реализация принтеров сильно различается, поэтому во FreeBSD существует большая библиотека драйверов для имеющихся моделей принтеров.

Драйверы принтеров функционируют независимо от технологии печати (лазерная, струйная и т.д.) и интерфейса устройства (параллельный порт, последовательный порт

RS-232, USB, сетевой принтер). Для драйверов FreeBSD не имеет значения, как именно формируется образ на бумаге. Интерфейс принтера более важен, поэтому он должен поддерживаться во FreeBSD. Параллельные и последовательные порты, а также протоколы сетевой печати хорошо поддерживаются, чего не скажешь о стандарте USB. Впрочем, если аппаратный интерфейс работает, то для драйвера принтера детали не имеют значения.

Во FreeBSD драйверы принтеров имеют кое-что общее с видеодрайверами XFree86: они являются частью отдельного программного пакета. Этот пакет называется *Ghostscript* (`http://www.cs.wisc.edu/~ghost/`) и отвечает за преобразование PostScript-файлов в формат, поддерживаемый принтером. Принципы печати во FreeBSD подробно описываются в главе 9. Там же рассказывается о том, почему язык PostScript является стандартом. А пока что достаточно лишь знать, что UNIX-программы, направляющие данные на печать, подготавливают их либо в обычном текстовом формате, либо в формате PostScript и помещают в очередь. Обработчик очереди передает PostScript-файлы программе Ghostscript, которая формирует двоичный файл, понятный принтеру. Это значит, что для FreeBSD лучше всего подходят принтеры, имеющие встроенную поддержку языка PostScript, так как отпадает необходимость в программе Ghostscript. PostScript-принтеры обычно стоят дороже, чем обычные принтеры. Если у вас уже есть старый принтер, проверьте, поддерживается ли он программой Ghostscript. Соответствующую информацию можно найти в базе данных Linux Printing Support Database по адресу `http://www.linuxprinting.org/printer_list.cgi`. Несмотря на название, информация в этой базе данных относится как к Linux, так и к FreeBSD.

Внимание

Некоторые принтеры рекламируются как PostScript-совместимые, хотя в действительности это не так: они контролируются программным обеспечением Windows, которое выполняет функции, схожие с Ghostscript. Такие принтеры бесполезны во FreeBSD. Иногда принтеры снабжаются интерпретатором языка PostScript, написанным не компанией Adobe (создателем языка), а кем-то другим. Обычно эти принтеры работают вполне нормально.

Если принтер не является PostScript-совместимым и не поддерживается программой Ghostscript, польза от него будет невелика. Иногда можно запустить программу Ghostscript (или другой интерпретатор PostScript) в Windows, подключить принтер к Windows-системе и использовать его как ресурс совместного доступа. Если же нужен непосредственный доступ, то лучше купить новый принтер. Программа Ghostscript поддерживает большинство современных принтеров. Проблемы возникают лишь с самыми дешевыми принтерами старых моделей.

Вспомогательные устройства

В настоящее время доступно множество других устройств помимо описанных выше. Многие из них требуют наличия специального драйвера. Всю необходимую информацию о них можно найти в файле `HARDWARE.TXT`. Перечислим наиболее важные устройства:

- **Звуковые платы**. Рынок звуковых плат очень разнообразен. Во FreeBSD поддерживаются многие популярные платы, но иногда не хватает поддержки менее популярных или новейших моделей. Если плата не поддерживается системой, лучше не оставлять ее в компьютере, чтобы не возникало проблем.
- **USB-устройства**. Все современные компьютеры оснащаются USB-портами. Поддержка этого стандарта во FreeBSD пока ограничена, хотя уже поддерживаются два основных микропроцессорных набора (известных как UHCI и

OHCI), а также ряд USB устройств. В отличие от некоторых внешних устройств, необходимо чтобы ядро FreeBSD имело средства поддержки конкретного USB-устройства или по крайней мере класса устройств (скажем, принтеры или модемы). Узнать текущую информацию о поддержке стандарта USB можно по адресу `http://www.etla.net/~n_hibma/usb/`. Чтобы использовать новейшие драйверы USB, ядро системы придется переконфигурировать (подробнее об этом рассказывается в главе 12).

- **Устройства со съемными носителями.** FreeBSD очень хорошо поддерживает дисководы гибких дисков и CD-ROM. Со съемными дисковыми накопителями такими как Zip, LS-120 или Jaz, можно работать примерно так же, как и с дискетами (некоторые из носителей разбиты на разделы подобно жесткому диску, что обычно не вызывает проблем). Съемные устройства с интерфейсом USB поддерживаются слабо. Устройства записи на компакт-диски и DVD-диски интерпретируются несколько необычно. Для них нужна специальная программа, которая будет подготавливать образ диска и "прожигать" его на носителе.
- **Сканеры.** Сканеры подключаются через параллельные порты, SCSI-интерфейс или порты USB. SCSI-сканеры поддерживаются лучше всего, сканеры с параллельным или USB-интерфейсом — хуже. Для каждой модели сканера нужен свой драйвер, являющийся частью пакета *SANE* (`http://www.sane-project.org`). Подобно пакетам XFree86 и Ghostscript, этот пакет реализован отдельно от ядра FreeBSD.
- **Устройства ввода.** Пользователи взаимодействуют с компьютером, вводя информацию посредством клавиатуры, мыши, иногда других устройств. Уже много лет клавиатуры персональных компьютеров стандартизированы, благодаря чему клавиатурные драйверы FreeBSD работают стабильно и практически незаметно для пользователя. Пакет XFree86 поддерживает различные типы мышей, включая все популярные программные протоколы. В настоящее время наиболее распространен протокол PS/2. В конце 90-х появился стандарт USB, поэтому начали набирать популярность клавиатуры и мыши с интерфейсом USB. Во FreeBSD они поддерживаются, но не так хорошо, как традиционные устройства.

Резюме

Прежде чем приступать к инсталляции FreeBSD, следует изучить возможности и специфические требования системы. Компьютер, работающий под управлением FreeBSD, может функционировать как рабочая станция или сервер. Вообще FreeBSD — популярная платформа для серверов Internet. FreeBSD способна взаимодействовать с другими операционными системами как по сети, так и в пределах одного компьютера. Мультисистемная среда может быть очень удобна, но для этого нужно определенным образом настроить каждую систему.

FreeBSD выдвигает не столь жесткие требования к аппаратной части компьютера, как современные версии Windows, более экономно расходуя ресурсы центрального процессора и оперативной памяти. Однако необходимо тщательно выбирать аппаратные компоненты, поскольку FreeBSD не так хорошо, как Windows, поддерживает некоторые вспомогательные устройства, в частности звуковые платы и даже принтеры.

Полный справочник по

Глава 2

Инсталляция системы

В главе 1 рассказывалось об истории FreeBSD, ее аппаратных требованиях, о том, какими преимуществами она обладает и как взаимодействует с другими операционными системами. В этой главе описывается процедура инсталляции системы и конкретизируются вопросы аппаратной поддержки.

Поскольку FreeBSD — открытая система, ее можно загрузить из множества источников. Об этом пойдет речь в начале главы. Далее рассказывается о подготовке компьютера к запуску FreeBSD. Вслед за этим рассматривается собственно процесс инсталляции. Указывается, какое программное обеспечение можно инсталлировать и как создать начальную учетную запись.

Получение FreeBSD

Те, кому не терпится установить систему, могут воспользоваться компакт-диском прилагаемым к книге. На нем содержится самая последняя версия системы, доступная на момент выпуска книги, — 5.1. Однако существуют и другие каналы получения системы, причем в более полном варианте, чем на компакт-диске.

Пользователи редко заботятся об условиях лицензирования системы, так как эти условия весьма просты и либеральны. Тем не менее знание их может оказаться важным в некоторых ситуациях, поэтому мы начнем главу с рассмотрения лицензионных требований.

Лицензирование

Многие считают, что FreeBSD распространяется на условиях *лицензии BSD*. Это очень короткая и простая лицензия. Она явным образом предоставляет всем желающим право использовать и распространять программное обеспечение, как в виде исходных кодов, так и в исполняемом формате, в исходном либо модифицированном варианте, но с одним условием: пометки об авторских правах и (для бинарного дистрибутива) отсутствии гарантий должны быть сохранены. Можно получить одну копию программного обеспечения, имеющего лицензию BSD, и инсталлировать ее на произвольное число компьютеров. При этом не имеет значения, было ли программное обеспечение загружено бесплатно через Internet, или вы заплатили тысячу долларов розничному продавцу (конечно, вряд ли кому-то удается продать свободно распространяемое программное обеспечение по такой цене, но некоторые организации, разрабатывающие программы с открытым исходным кодом, продают компакт-диски со своими программами, чтобы собрать средства на дальнейшую разработку).

Так можно ли сказать, что на операционную систему FreeBSD распространяются условия лицензии BSD? Лишь частично. Ядро системы и многие служебные программы действительно имеют эту лицензию. Но есть и другие. Например, пакет XFree86 имеет лицензию МТИ (Массачусетского технологического института), которая очень напоминает лицензию BSD; компилятор GCC имеет GNU-лицензию *GPL* (General Public License). Последняя столь распространена, что заслуживает отдельного упоминания. Это довольно большая лицензия, налагающая ограничения на распространение программ с открытым исходным кодом. Она, в частности, требует, чтобы модифицированные программные продукты распространялись на тех же условиях, т е с лицензией GPL. В лицензии BSD о производных лицензиях ничего не говорится, и это позволяет дистрибьютору выбирать для модифицированного продукта любую лицензию. Авторы GPL заявляют, что их подход лучше, так как гарантируется, что производные продукты будут доступны с открытым исходным кодом (или бесплатно, если хотите). У авторов BSD своя точка зрения: они считают, что их лицензия лучше, по

скольку позволяет использовать их программные продукты как угодно, даже создавать на их основе коммерческие системы Обе стороны заявляют, что их лицензии обеспечивают больше свободы, но понятие "свобода" каждая сторона трактует по-своему За философским диспутом между сторонниками лицензий BSD и GPL следит все сообщество разработчиков программного обеспечения с открытым исходным кодом.

На заметку

Термин "*открытый исходный код*" имеет точное определение Подробности можно узнать на Web-узле `www.opensource.org` На Web-узле FreeBSD приведены ссылки на важные лицензионные соглашения, используемые системой (`http //www freebsd.org/copyright/`)

GPL MIT и другие лицензии на программы с открытым исходным кодом дают пользователям почти одинаковые права. Они позволяют загружать или другим способом получать программу и устанавливать ее на произвольное число компьютеров Можно создавать копии инсталляционного носителя и передавать их друзьям и коллегам, более того — даже продавать их. Но вот если вы захотите модифицировать программу, изучите лицензионные требования чтобы случайно не нарушить какие либо ограничения

Большинство программ FreeBSD имеет ту или иную открытую лицензию. Некоторые программы из *коллекции переносимых приложений FreeBSD* (набор программ, легко инсталлируемых во FreeBSD, но официально не являющихся частью системы как таковой) используют лицензии иного типа В таких случаях нужно внимательно следить за соблюдением лицензионных требований Чаще всего лицензия ограничивает распространение программы на носителях, предназначенных для продажи. Подробности можно узнать по адресу `http://www.freebsd.org/copyright/LEGAL` Даже некоторые пакеты самой FreeBSD с технической точки зрения не являются программами с открытым исходным кодом, хотя разрешается их бесплатное распространение.

Подводя итог вышесказанному, отметим, что FreeBSD представляет собой смесь программ с различными лицензиями Большую часть из них можно классифицировать как программы с открытым исходным кодом Образы компакт-дисков FreeBSD, доступные в Internet, и сами компакт-диски, приобретаемые у продавцов, допускают копирование и инсталляцию на произвольном числе компьютеров Лишь некоторые программные пакеты имеют более жесткие ограничения. Подробная информация об этом содержится в файле `README`, `LICENSE` или `COPYRIGHT` на компакт-диске.

Источники

FreeBSD может быть получена разными способами Перечислим наиболее распространенные из них

- **Пакетный набор** Некоторые компании продают дистрибутивные компакт диски FreeBSD в комплекте с документацией, дополнительным программным обеспечением и другими компонентами. В некотором смысле эта книга представляет собой как раз такой набор. Но она имеет ценность именно как книга. Другие наборы содержат больше программного обеспечения, сопровождаемого печатной документацией, которую можно найти в электронном виде В любом случае такие наборы полезны для тех, кто наряду с традиционным программным обеспечением хочет получить печатную документацию

На заметку

К книге прилагается мини-версия FreeBSD, помещающаяся на одном компакт-диске. Если нужна более полная версия, то можно загрузить недостающие компоненты из Internet или поискать полную версию на нескольких компакт-дисках или DVD-диске.

- **Дистрибутивный компакт диск**. Можно купить компакт-диск набор компакт дисков или DVD-диск, содержащий FreeBSD, без какой-либо печатной документации Это обойдется дешевле пакетного набора, но отсутствие печатной документации может оказаться недостатком. Такой вариант предпочтителен или опытных пользователей, а также в том случае, когда нужно получить самую последнюю версию программного обеспечения в дополнение к уже имеющей ся документации по FreeBSD
- **FTP-узлы**. FreeBSD доступна в Internet, как правило, на FTP-узлах При нали чии высокоскоростного соединения можно загрузить весь системный образ и даже записать его на компакт-диск. В качестве альтернативы можно загрузить лишь образ начальной дискеты, а затем инсталлировать FreeBSD непосредственно с FTP-узла на жесткий диск Список *FTP-зеркал* – узлов, хранящих копии официального FTP-узла FreeBSD, – доступен по адресу `http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/mirrors-ftp.html` Выберите ближайшее "зеркало", чтобы ускорить процесс загрузки

Если вы предпочитаете самостоятельно записывать компакт-диски, загрузите с FTP-узла *файлы образов* CD-R (файлы с побайтовыми копиями содержимого записываемого компакт-диска) и перенесите их на компакт-диск формата CD-R. Эти файлы можно найти на официальных FTP-узлах FreeBSD Ссылки на них содержатся также на Web-узле `www.linuxiso.org` Создаваемый таким способом компакт-диск функционально идентичен магазинному варианту Установка по сети, при которой программа инсталляции читает файлы с удаленного сервера, удобна при наличии высокоскоростного сетевого соединения

Если нужно установить FreeBSD на нескольких компьютерах локальной сети создайте собственный сервер инсталляций, скопировав содержимое дистрибутивного компакт-диска яяи соответствующего каталога FTP узла на жесткий диск сервера (можно выполнять сетевую инсталляцию непосредственно с компакт диска но это более медленный способ) Затем запустите программу сетевой инсталляции и нацельте ее на сервер. Сервер, хранящий файлы FreeBSD, может работать под управлением любой операционной системы, но на нем должно выполняться серверное программное обеспечение FTP или NFS

Подготовка инсталляционного носителя

Если был приобретен коробочный набор FreeBSD, то все необходимые инсталляционные носители, скорее всего, уже имеются В остальных случаях придется подготовить их Процесс подготовки состоит из двух этапов 1) создание загрузочного диска; 2) создание источника инсталляции Загрузочным носителем может быть дискета, компакт-диск или другое устройство Исходные файлы обычно находятся на загрузочном компакт-диске, но если загрузка осуществляется с дискеты, то они расположены на отдельном компакт-диске локальном или серверном жестком диске.

Загрузочные диски

Большинство продаваемых сегодня компакт дисков с FreeBSD являются загрузочными Современный компьютер загружается с такого компакт-диска и немедленно запускает программу инсталляции FreeBSD (для этого может понадобиться поменять некоторые настройки BIOS) Если компакт-диск создавался на основе файла образа, то он является загрузочным. Если же на него помещались отдельные файлы, загруженные из Internet, или компьютер не поддерживает загрузку с компакт-диска придется создать другой загрузочный носитель для запуска инсталлятора FreeBSD

Чаще всего выбор делается в пользу дискеты На компакт диске и FTP узле FreeBSD содержатся образы загрузочных дискет Они называются kern.flp и mfsroot.flp и находятся в каталоге floppies. Если на компьютере установлена DOS или Windows, перейдите в указанный каталог компакт-диска, вставьте чистую отформатированную дискету в дисковод A: и введите следующие команды (перед второй командой нужно вставить новую дискету):

```
E:> ..\TOOLS\FDIMAGE KERN FLP A:
E:> ..\TOOLS\FDIMAGE MFSROOT.FLP A:
```

Эти команды запускают DOS-программу fdimage.exe, которая заполняет дискету по предоставленному образцу. Первая дискета (kern.flp) предназначена для загрузки системы, а вторая содержит инсталляционные утилиты Если у вас есть доступ к компьютеру, на котором установлена FreeBSD или другая UNIX-система, аналогичные действия можно выполнить с помощью стандартной UNIX-команды dd. Например если компакт-диск смонтирован в каталоге /cdrom необходимо ввести следующие команды:

```
# dd if=/cdrom/floppies/kern.flp of=/dev/fd0
# dd if=/cdrom/floppies/mfsroot flp of=/dev/fd0
```

Идентификатор дисковода гибких дисков (/dev/fd0) и точка монтирования компакт-диска могут меняться, но в большинстве UNIX систем команды именно таковы

Источник инсталляции

Если система устанавливается с компакт-диска купленного в магазине или с официального FTP-узла то подготавливать инсталляционный носитель не нужно Если же загружается образ записываемого компакт диска, то нужно перенести его на компакт-диск формата CD-R. Как именно это делается, зависит от используемого программного обеспечения Во FreeBSD и других UNIX-системах можно воспользоваться утилитой cdrecord:

```
# cdrecord dev=6,0 speed=4 freebsd.iso
```

Номера устройства (dev=6,0), скорость записи и название файла образа зависят от системы.

В Windows или Mac OS следует поискать опцию "записывать компакт диск по файлу образа", "записывать компакт-диск по образу ISO" или что-то в этом роде. Это необходимо, чтобы на компакт-диске не создавалась файловая система, как обычно происходит в программах записи компакт-дисков для Windows и Mac OS.

Если при записи компакт-диска воспользоваться установками по умолчанию, то будет получен компакт-диск, содержащий единственный файл образа, к тому же не являющийся загрузочным. Такой компакт диск бесполезен для инсталляции FreeBSD Корректно записанный компакт-диск содержит в корневом каталоге несколько файлов и подкаталогов, в том числе tools и floppies

Подготовка компьютера к запуску FreeBSD

Прежде чем инсталлировать FreeBSD, необходимо подготовить компьютер к запуску операционной системы. Первый этап подготовки — проверка совместимости оборудования с FreeBSD. Если какие-то устройства не поддерживаются, нужно либо поменять их, либо загрузить обновленные драйверы FreeBSD. Второй этап — очистка дискового пространства под операционную систему. Этот этап упростится, если на

компьютере предполагается использовать только FreeBSD Если же на компьютере уже установлена другая операционная система, может потребоваться добавить жест кий диск, уменьшить существующие разделы или изменить их конфигурацию

Проверка аппаратной совместимости

Основные вопросы аппаратной совместимости уже рассматривались в главе 1 Среди базовых компонентов системы требуют проверки SCSI-адаптеры, сетевые и графические платы. Для них нужны специальные драйверы FreeBSD (или драйверы программных компонентов FreeBSD, таких как XFree86). В основном эти устройства хорошо поддерживаются системой, но бывают и исключения. Менее важные устрой ства, в частности звуковые платы, платы видеомонтажа, внутренние модемы, сканеры и принтеры, тоже требуют наличия драйверов Здесь выбор не столь велик, хотя некоторые типы устройств, например принтеры и звуковые платы, поддерживаются достаточно хорошо Вся необходимая информация по этому поводу содержится в системном файле `HARDWARE.TXT` (он также доступен по адресу `ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/5 0 RELEASE/`).

Работа некоторых устройств четко стандартизирована, поэтому для них использу ются универсальные, широко распространенные драйверы. К таким устройствам от носятся центральные процессоры (семейство x86), материнские платы, последователь ные порты RS-232, параллельные порты, порты клавиатуры, порты мыши (PS/2), дисководы гибких дисков, определенные модели жестких дисков и дисководов CD-ROM, а также ряд контроллеров EIDE. Последние работают одинаково лишь в низкопроизводительных режимах, и для повышения производительности требуются специализированные драйверы Поскольку контроллеры EIDE интегрированы в современные материнские платы, этот вопрос обычно рассматривают применительно к последним, но в худшем случае может потребоваться добавить отдельную плату контроллера, если встроенный контроллер не поддерживает ускоренные режимы

Если на компьютере в настоящий момент установлена Windows, то с помощью па нели управления можно многое узнать о его аппаратном обеспечении. На рис. 2 1 представлено окно диспетчера устройств (из Windows 2000) с информацией о присут ствующих устройствах Здесь можно выяснить название набора микросхем и файла драйвера, а также узнать, кто является изготовителем устройства Но эта информация не должна вводить в заблуждение К примеру, название изготовителя может не играть никакой роли, поскольку большинство драйверов FreeBSD пишется для конкретных *микропроцессорных наборов*, используемых устройством В такой набор входит одна или несколько микросхем, реализующих функции устройства Многие производители ис пользуют микропроцессорные наборы других производителей

Еще один способ получить информацию об устройствах — посмотреть на сами микросхемы. Ищите самую крупную микросхему на плате. На нее обычно нанесено название изготовителя или его логотип вместе с номером модели. На рис 2 2 изображена главная микросхема видеоадаптера Matrox Millennium. По ее маркировке понятно, что на плате установлен микропроцессорный набор Matrox MGA 2064W Эту ин формацию возможно, придется указать при инсталляции системы или ее последую щем конфигурировании

Внимание! Некоторые микросхемы, например на многих современных графиче ских и материнских платах, генерируют столько тепла, что помещены в специальные радиаторы, которые скрывают их маркировки *Не пытайтесь* снять такой радиатор, поскольку можно повредить микросхему Но даже если вы все сделаете аккуратно, придется потратить немало времени на то чтобы потом установить радиатор обратно

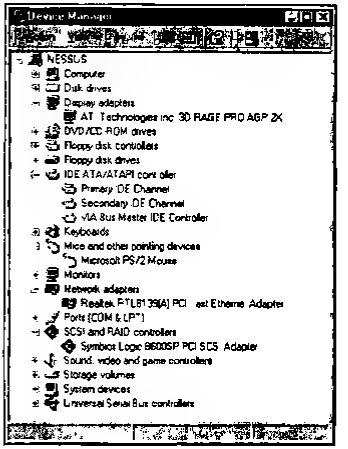


Рис 2 1 Окно диспетчера устройств Windows позволяет собрать информацию о компьютере перед инсталляцией FreeBSD

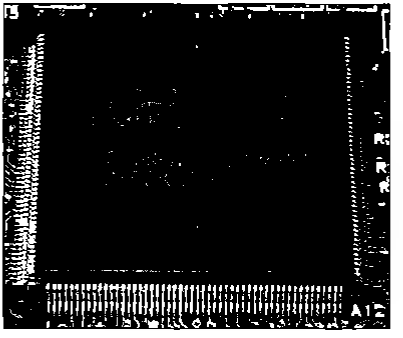


Рис 2 2 Микросхемы устройств обычно помечают ся для упрощения идентификации

Если устройство не упомянуто в файле `HARDWARE.TXT`, это не означает, что оно не работает Бывает, что устройство известно под несколькими названиями, и в файле указано только одно из них Можно поискать информацию об устройстве в Internet или просто инсталлировать FreeBSD система автоматически обнаруживает многие устройства, так что все быстро станет на свои места

Если устройство все же не работает, существуют три возможных решения

- **Замена**. Замените нерабочее устройство эквивалентной моделью, которая поддерживается системой. Чаще всего это наиболее эффективный и простой способ решения проблемы, особенно если речь идет о недорогих устройствах, таких как сетевые и звуковые платы. Многие компоненты материнских плат нельзя удалить, но можно добавить плату расширения, которая продублирует требуемую функцию. В крайнем случае отключите этот компонент в BIOS.
- **Настройка**. Существуют способы устранения несовместимости без замены устройства. Например, устройство может работать в режиме с ограниченными функциональными возможностями, и этого вполне достаточно. Некоторые принтеры поддерживают режим низкого разрешения, доступный во FreeBSD (точнее, в пакете Ghostscript), а режимы с высоким разрешением остаются недоступными. Иногда можно вообще обойтись без устройства. Скажем, если звуковая плата используется в основном для игр в Windows, то во FreeBSD ее лучше отключить.
- **Разработка драйвера**. Опытные программисты могут написать нужный драйвер самостоятельно. Рассмотрение этого вопроса выходит за рамки книги, поскольку такое решение оправданно лишь в том случае, когда стоимость замены компонента огромна (например, в компьютерах крупной организации установлено несколько сот дорогих плат, которые не поддерживаются в новой системе).

В некоторых случаях для аппаратного устройства имеется поддержка, но лишь в виде экспериментального драйвера или программного обеспечения, не входящего в состав FreeBSD. Если устройство играет важную роль в обеспечении работы компьютера (например, SCSI-адаптер, управляющий жесткими дисками), то перед установкой системы придется создать специальную инсталляционную дискету. Что касается менее важных устройств, таких как звуковые платы или принтеры, то можно инсталлировать FreeBSD, а новый драйвер добавить позднее. Подробная информация об этом должна содержаться в документации к драйверу.

Очистка дискового пространства

Системе для работы необходимо место на диске. Если FreeBSD инсталлируется на новый компьютер или поверх другой операционной системы, то о выделении дискового пространства можно не беспокоиться. Как будет описано ниже, достаточно предложить инсталлятору использовать все дисковое пространство, и он удалит все имеющиеся на диске данные.

Ситуация усложняется, если требуется добавить FreeBSD к другой операционной системе. В этом случае необходимо выделить под FreeBSD первичный раздел. На многих дистрибутивных компакт-дисках FreeBSD записана открыто распространяемая программа FIPS (`http://www.igd.fhg.de/~aschaefe/fips/`). Она позволяет разбить существующий первичный раздел FAT на две части. Затем с помощью инсталлятора FreeBSD можно удалить второй раздел и использовать его пространство. Существуют и другие аналогичные программы сторонних разработчиков, например PartitionMagic компании PowerQuest (`www.powerquest.com`). Такие программы обычно проще в использовании, чем FIPS. Очень важным преимуществом PartitionMagic является то, что эта программа распознает разделы NTFS, широко используемые в Windows NT/2000/XP.

FIPS, PartitionMagic и другие утилиты динамической разбивки на разделы небезопасны. Ошибки в исполняемом коде, сбои питания и другие проблемы могут привести к потере данных и даже к разрушению ключевых структур файловой системы, что сделает доступ к диску невозможным без сложной и затратной процедуры восстановления. По этой причине рекомендуется осуществлять резервное копирование данных перед запуском любой из описываемых ниже программ.

Программа FIPS

Чтобы с помощью программы FIPS уменьшить существующий раздел FAT, следуйте приведенной ниже процедуре

1 В DOS или Windows перейдите на диск C: и запустите программу дефрагментации диска. В Windows кнопка запуска дефрагментатора доступна в окне свойств диска (рис. 2.3). Идея заключается в том, чтобы переместить все файлы в начало диска, оставив в конце пустое место, которое и будет отведено под FreeBSD. Желательно также проверить целостность диска. Учтите, что некоторые утилиты записывают в конец диска файлы, которые запрещается перемещать. Такие утилиты нужно временно отключить, удалив их файлы данных

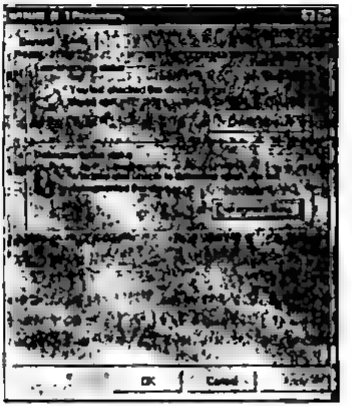


Рис. 2.3. В окне свойств доступна кнопка дефрагментации диска

2 Загрузитесь в DOS. Убедитесь в том, что на загрузочной дискете есть драйверы CD-ROM. Если их нет, найдите в каталоге `TOOLS` дистрибутивного комплекта диска FreeBSD программу `FIPS.EXE` и скопируйте ее на дискету или в раздел жесткого диска, доступный в DOS. Вместо DOS можно воспользоваться FreeDOS (`www.freedos.org`) — реализацией DOS с открытым исходным кодом.

Внимание!

Не запускайте программу FIPS в многозадачной операционной системе, такой как OS/2 или Windows (в Windows 9x загрузитесь в режиме командной строки, вместо того чтобы запускать программу в DOS окне). Это вызовет сбой операционной системы и приведет к повреждению файлов

3 Перейдите в каталог, где находится программа FIPS, и запустите ее. Появится приветственное сообщение.

4 Нажмите любую клавишу. Программа отобразит информацию о таблице разделов жесткого диска. Эта информация относится только к первичным разделам, поскольку с логическими разделами программа не работает. Границы разделов определяются начальными и конечными номерами цилиндров, головок и секторов

Эти номера используются в BIOS для адресации диска. Путем модификации номера цилиндра происходит уменьшение раздела DOS или Windows

5. Нажмите любую клавишу. Программа отобразит информацию о разделе C:. Проверьте правильность сведений о структурах данных FAT. Программа спросит, нужно ли создавать резервную копию корневого и загрузочного разделов. Желательно ответить y, чтобы можно было отменить операцию в случае необходимости.

6. Программа спросит, находится ли загрузочная дискета в дисководе A:. Вставьте дискету и нажмите y. Программа запишет на дискету содержимое корневого и загрузочного разделов.

7. Программа вычислит минимальный размер, который может иметь существующий раздел C:, и отобразит его вместе с размером нового первичного раздела. Если нужно увеличить раздел C: (чтобы предусмотреть место для новых файлов DOS или Windows), нажимайте клавишу <→> до тех пор, пока не будет достигнут требуемый размер. Приращение происходит по одному цилиндру за раз и осуществляется за счет сокращения раздела FreeBSD. По окончании нажмите <Enter>.

8. Программа отобразит итоговую информацию о новой структуре разделов в том виде, как это было в п. 4. Если вас все устраивает, нажмите e, чтобы программа начала вносить изменения. В противном случае нажмите r и повторите п. 7.

9. Программа отобразит итоговую информацию об обновленном разделе C:, как в п. 5. Для продолжения нажмите y, после чего программа запишет изменения на диск.

С этого момента на жестком диске два первичных раздела, а не один. При установке FreeBSD можно попросить инсталлятор удалить второй раздел и создать вместо него первичный раздел FreeBSD.

К сожалению, возможности программы FIPS ограничены. Она не умеет менять размеры логических разделов и разделов NTFS, перемещать разделы и выполнять ряд других полезных действий. Для этого нужна более сложная программа, такая как PartitionMagic.

Программа PartitionMagic

Коммерческая программа PartitionMagic обладает более широкими возможностями, чем FIPS. С ее помощью можно создавать, удалять, перемещать и модифицировать разделы FAT, NTFS, HPFS и ext2fs (поддержка HPFS была удалена в версии 7.0, файловая система ext2fs, используемая в Linux, поддерживается начиная с версии 4.0). Перед запуском программы не нужно выполнять дефрагментацию диска, хотя проверить его целостность не помешает.

Программа PartitionMagic может запускаться как с загрузочной дискеты DOS, так и в среде Windows. Интерфейс программы в обоих случаях примерно одинаков, хотя DOS-версия менее функциональна (в частности, программа не умеет автоматически выбирать схему разбивки на разделы для различных комбинаций операционных систем). Программа отображает графический образ диска в виде диаграммы разделов, а также выдает текстовый список разделов. Щелкните на названии раздела в любой из областей и выберите команду Partitions⇨Resize, чтобы изменить его размер. Появится диалоговое окно, в котором можно ввести размер нового раздела в числовом виде или воспользоваться специальным ползунковым регулятором в верхней части окна (рис. 2.4).

Проще всего оставить дисковое пространство, отведенное под FreeBSD, не инициализированным. Программа PartitionMagic не умеет создавать первичные разделы FreeBSD — это придется делать с помощью инсталлятора FreeBSD. Главное, чтобы выделенное пространство не находилось в расширенном разделе, так как в этом случае оно недоступно для FreeBSD.

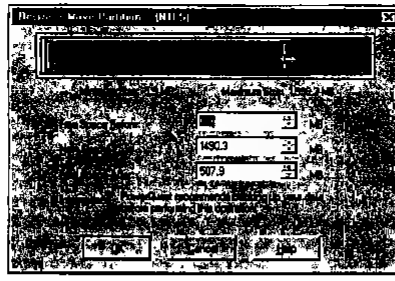


Рис. 2.4. Программа PartitionMagic позволяет менять размер раздела с помощью графического регулятора

Задав необходимые размеры, щелкните на кнопке Apply в основном окне программы. Программа зафиксирует сделанные изменения, и выделенное пространство станет доступно для FreeBSD.

Начало инсталляции

Выделив для FreeBSD дисковое пространство, можно приступать к инсталляции системы. Начальный этап включает в себя загрузку инсталлятора FreeBSD, установку начальных конфигурационных параметров и создание дисковых разделов.

Загрузка инсталлятора

Инсталлятор — это специальный усеченный вариант FreeBSD, включающий программы копирования системных файлов с инсталляционного носителя на жесткий диск. Поддерживаются два типа загрузочных носителей:

- **Компакт-диски**. Большинство дистрибутивных компакт-дисков FreeBSD, включая прилагаемый к этой книге, являются загрузочными. Предварительно необходимо сконфигурировать BIOS так, чтобы поиск системного загрузчика выполнялся сначала на компакт-диске и лишь затем — на жестком диске. Если загрузка с компакт-диска не поддерживается в BIOS или по каким-то причинам недоступна, воспользуйтесь дискетами.
- **Дискеты**. На с. 58 рассказывалось о том, как создать инсталляционные дискеты. Первая из них является загрузочной. Сконфигурируйте BIOS так, чтобы поиск системного загрузчика велся в первую очередь на дискете, после чего вставьте загрузочную дискету и перезапустите компьютер. Проинициализировав систему, загрузчик попросит вставить вторую дискету, с которой будет запущена программа инсталляции FreeBSD.

Процедура инсталляции FreeBSD не зависит от того, загружается ли инсталлятор с компакт-диска или дискеты. В обоих случаях на экране появится множество сообщений, связанных с внутренними проверками ядра и инсталляцией аппаратных драйверов. Как правило, на них можно не обращать внимания.

Далее появится окно программы `sysinstall` (рис. 2.5). Это и есть инсталлятор FreeBSD. Программу `sysinstall` можно запускать и по окончании инсталляции для

добавление или удаление программных компонентов, а также для обновления операционной системы.

Пользовательский интерфейс программы требует некоторых разъяснений. Для выбора элементов меню используются клавиши управления курсором. Клавиши <↑> и <↓> предназначены для перемещения по списку элементов (Usage, Standard и т.д. на рис. 2.5), а клавиши <←> и <→> — для выбора командных кнопок в нижней части окна (<Exit> и <Help>, на рис. 2.5). Клавиши <Page Up> и <Page Down> позволят прокручивать длинные списки, не помещающиеся на экране. С помощью клавиши <Пробел> можно устанавливать и сбрасывать флажки — элементы меню рядом с которыми стоят квадратные скобки (на рис. 2.5 таковых нет). Если в квадратных скобках присутствует значок X, значит, флажок установлен. Для выбора активной командной кнопки нужно нажать <Enter>. В некоторых меню эта клавиша эквивалентна пробелу. В большинстве элементов меню начальные символы выделены другим цветом. Чтобы активировать такой элемент, достаточно нажать выделенную клавишу в сочетании с клавишей <Alt>. Например, чтобы начать инсталляцию в режиме Express, нажмите <Alt+E>.

FreeBSD поддерживает три типа инсталляции: стандартную (Standard), ускоренную (Express) и настраиваемую (Custom). Стандартный режим удобен для большинства случаев. В ускоренном режиме некоторые меню не отображаются и пропускается ряд проверок, поэтому инсталляция проходит быстрее, но впоследствии можно потратить больше времени на корректное конфигурирование системы. Настраиваемый режим предназначен для опытных пользователей, которые заранее знают, как настроить параметры. Далее предполагается, что инсталляция ведется в стандартном режиме.

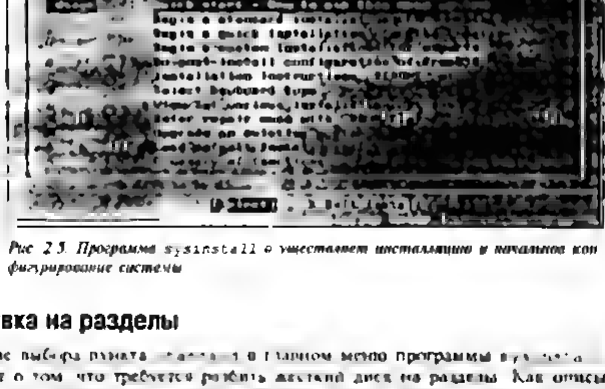


Рис. 2.5. Программа sysinstall в ожидании инсталляции и начальной конфигурирования системы

Разбивка на разделы

После выбора пункта Standard в главном меню программы установки система сообщит о том, что требуется разбить жесткий диск на разделы. Как описывалось в главе 1, это двухэтапный процесс. Сначала нужно создать стандартный раздел x86 (основной раздел FreeBSD) а затем в нем — BSD подразделы. Начальное сообщение говорит о том, что сейчас первый раздел. Выберите Ok, после чего появится окно программы FDISK Partition Editor (рис. 2.6).

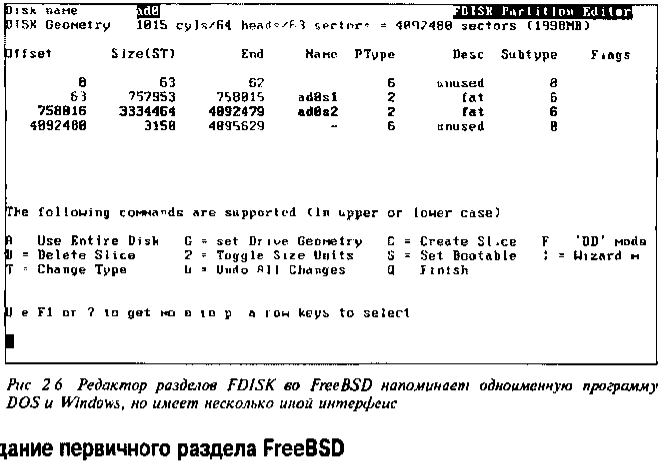


Рис 2 6 Редактор разделов FDISK во FreeBSD напоминает одноименную программу DOS и Windows, но имеет несколько иной интерфейс

Создание первичного раздела FreeBSD

Далее можно пойти двумя путями в зависимости от того как распределяется пространство на диске.

- **Использование всего диска** Если под FreeBSD нужно отвести весь диск, нажмите A, чтобы уничтожить все существующие разделы и данные Будет создан единый первичный раздел x86, целиком занимаемый FreeBSD
- **Создание новых разделов** Если на диске есть свободное место (помечается как unused в колонке Desc), нажмите C, чтобы создать первичный раздел FreeBSD ("slice") Система спросит, каков размер нового раздела Чаще всего под FreeBSD отводится все свободное пространство, поэтому достаточно нажать <Enter>, чтобы принять установку по умолчанию В противном случае нужно задать размер в виде количества 512-байтовых блоков либо в мегабайтах, добавив к числу суффикс M (например, 500M) Если от прежних инсталляций остался раздел FreeBSD, нужно выбрать и удалить его, нажав D а затем заново проинициализировать освободившееся дисковое пространство

На заметку Нередко можно видеть, что в начале и конце диска есть небольшие блоки неиспользуемого пространства, как на рис 2 6 Это следствие применения старого метода адресации разделов по номерам цилиндров, головок и секторов Такие блоки нельзя использовать сами по себе, но если рядом с одним из них создается раздел FreeBSD, то можно подключить неиспользуемое пространство, удалив раздел, отводимый под FreeBSD а затем воссоздав его с помощью команды C

Теоретически разрешается создавать несколько первичных разделов FreeBSD однако на практике необходимость в этом возникает редко Одно из исключений — когда в системе несколько жестких дисков

Обращайте внимание на названия первичных разделов На рис. 2 6 раздел ad0s1 содержит Windows, а раздел ad0s2 предназначен для FreeBSD (он будет удален и создан заново). Названия не нужно вводить вручную но они могут понадобиться при выполнении различных административных задач

Задание параметров загрузчика

Закончив вносить изменения в таблицу разделов нажмите Q. Все изменения будут записаны на диск, и система начнет задавать вопросы по поводу расположения загрузчика. На выбор предоставляются три опции:

- **BootMgr**. Загрузчик FreeBSD записывается в *главную загрузочную запись* (ГЗЗ) диска. Это означает, что он будет запускаться первым при включении компьютера. Такой вариант хорош, когда FreeBSD — единственная операционная система, установленная на компьютере. Если на компьютере несколько операционных систем и уже используется другой загрузчик, рассмотрите другие варианты.
- **Standard**. В ГЗЗ записывается стандартный загрузчик, запускающий систему только из активного раздела. Этот вариант можно использовать, если впоследствии планируется инсталлировать еще один загрузчик.
- **None**. Содержимое ГЗЗ не меняется. Это наилучший вариант, если уже инсталлирован другой загрузчик, например System Commander или LILO. Чтобы FreeBSD могла загружаться в дополнение к другим операционным системам, имеющийся загрузчик придется переконфигурировать.

Подробнее эта тема рассматривается в главе 4.

Создание подразделов

На втором этапе первичный раздел FreeBSD делится на подразделы или просто *разделы*, с помощью программы FreeBSD Disklabel Editor (рис. 2.7). Каждой системе необходимы минимум два раздела: корневой (/) и раздел подкачки. Если на компьютере установлена DOS или Windows, программа отобразит информацию о соответствующем первичном разделе.

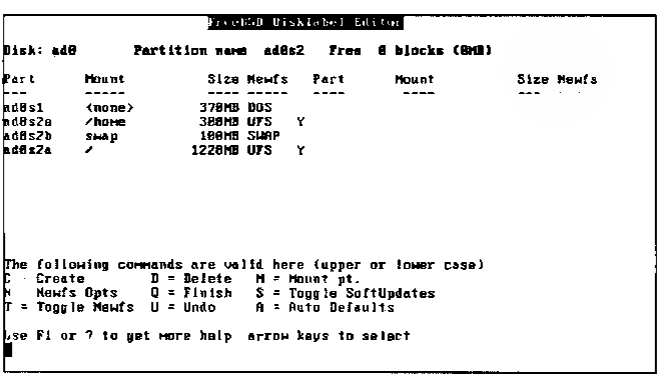


Рис 2 7 Программа FreeBSD Disklabel Editor позволяет разбивать первичный раздел на подразделы и назначать им точки монтирования

Убедитесь в том, что строка Disk в верхней части экрана выделена, и нажмите C для создания нового раздела. Программа попросит ввести следующую информацию:

1 Размер раздела (в блоках, мегабайтах или цилиндрах)

2. Тип содержимого раздела· файловая система или область подкачки Должен быть создан хотя бы один раздел второго типа Желательно, чтобы его размер был не меньше, чем объем оперативной памяти компьютера, так как в противном случае существует вероятность того, что при некоторых типах системных сбоев в разделах, расположенных после раздела подкачки, будут уничтожены данные

3 *Точка монтирования* раздела, т е. название каталога, через который будет доступно содержимое раздела У раздела подкачки точка монтирования отсутствует

Если на диске имеется несколько сот мегабайтов свободного дискового пространства, нажмите **A**, и система разделит это пространство в соответствии со стандартными правилами. Это удобно для тех, кто не знаком с FreeBSD. Более опытные пользователи предпочитают самостоятельно создавать разделы Ниже описаны разделы создаваемые по команде **A**

- **Корневой (/)** Это начало дерева каталогов FreeBSD Доступ ко всем файлам осуществляется именно через корневой каталог По умолчанию он делается маленьким, поскольку большинство файлов располагается в подразделах. На рис 2 7 это правило не соблюдено, так как в системе отсутствуют некоторые из перечисленных ниже разделов
- **/home.** В каталоге /home хранятся пользовательские файлы данных Следовательно, его оптимальный размер зависит от числа пользователей системы, а также их индивидуальных потребностей. В случае однопользовательской рабочей станции либо сервера с небольшим числом реально регистрирующихся пользователей, под каталог /home достаточно отвести несколько мегабайт или вообще включить его в корневой раздел В многопользовательской системе корневой каталог может занимать несколько гигабайт.
- **/usr** Каталог /usr традиционно содержит программные файлы Вот почему в стандартной конфигурации это самый большой раздел (за исключением, в некоторых случаях, каталога /home). Желательно выделить под этот каталог не менее 1 Гбайт (в минимальном варианте системе достаточно 100 Мбайт) Если каталог /usr не занимает отдельный раздел его размер нужно добавить к размеру корневого каталога.
- **/tmp.** В этом каталоге хранятся временные файлы периодически удаляемые специальными системными утилитами Под него отводится отдельный раздел, чтобы временные файлы не переполняли другие важные разделы
- **/var.** Этот каталог напоминает /tmp тем, что хранит преимущественно временные файлы Но они, как правило, создаются не пользовательскими приложениями, а самой системой и ее утилитами Например, здесь хранятся почтовые сообщения, пока они не будут прочитаны пользователями Как и /tmp, каталог /var часто помещают в отдельный раздел, чтобы защитить другие разделы от переполнения. Обычно размер раздела не превышает 1 Гбайт, но некоторые системы, например крупные почтовые серверы, нуждаются в гораздо большем хранилище

Правильно подобрать размеры разделов – одна из самых сложных задач для администраторов-новичков Если система является типичной рабочей станцией или сервером, через который проходит не очень много данных, то установки по умолчанию выбираемые с помощью команды **A**, вполне подойдут В случае сомнений попробуйте упростить конфигурацию, создав с помощью команды **C** только корневой раздел (/) каталог /home и раздел подкачки. Такая конфигурация чревата тем, что неуправляемый процесс может переполнить корневой раздел временными файлами Но это все-таки довольно редкая ситуация Гораздо неприятнее постоянно испытывать нехватку

места для временных файлов в слишком маленьких разделах /var и /tmp. В то же время опытный пользователь точно знающий как будет использоваться система, может создать дополнительные разделы для таких каталогов, как, например, /opt или /usr/X11R6.

В большинстве систем раздел подкачки используется очень интенсивно, поэтому его нередко помещают между обычными разделами. Это сокращает время перемещения между разделами и способствует повышению производительности системы.

Когда разделы будут созданы, нажмите Q для перехода к следующему этапу инсталляции.

Выбор инсталлируемого программного обеспечения

При выборе инсталляционной среды программа sysinstall отображает список дистрибутивных наборов, доступных для установки (рис. 2.8). Каждый набор представляет собой коллекцию взаимосвязанных программных пакетов, например средств разработки или пользовательских приложений. Можно выбрать конкретный набор или группу наборов либо инсталлировать все наборы, воспользовавшись командой All. Для каждого набора программа спрашивает, следует ли инсталлировать коллекцию переносимых приложений, т.е. программных "адаптеров", позволяющих компилировать приложения из исходных кодов (не все программы доступны в виде исполняемых файлов, хранятся скомпилированные под FreeBSD). По завершении выбора можно воспользоваться командой Exit для возврата к предыдущему меню (рис. 2.8) или продолжить инсталляцию.

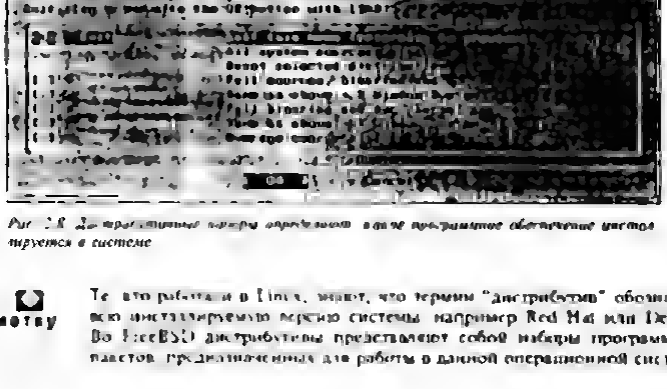


Рис. 2.8. Дистрибутивные наборы определяют какое программное обеспечение инсталлируется в системе

На заметку

Те, кто работали в Linux, знают, что термин "дистрибутив" обозначает всю инсталлируемую версию системы, например Red Hat или Debian. Во FreeBSD дистрибутивы представляют собой наборы программных пакетов, предназначенных для работы в данной операционной системе.

Если на компьютере должна быть установлена графическая среда, выберите дистрибутив `X-User`. На практике для большинства систем необходим и один из дистрибутивов для разработчиков. Дело в том, что большая часть программного обеспечения UNIX доступна в виде исходных кодов, поэтому, даже если вы не программист, средства разработки понадобятся вам для компиляции программ, которые не являются официальными исполняемыми компонентами FreeBSD (подробнее об этом рассказывается в главе 11). Важным дистрибутивом для разработчиков является `Kern-Developer` (и его вариант `X-Kern-Developer`). Он содержит средства, позволяющие модифицировать и компилировать ядро FreeBSD (см. главу 12). Если какой-то дистрибутив не был выбран, а впоследствии оказывается, что он необходим, инсталлируйте его, повторно запустив программу `sysinstall`.

Тот, кто хорошо знаком со структурой программных пакетов FreeBSD, может выбрать режим `Custom`. В этом случае программа позволит включать и отключать конкретные дистрибутивы. (В меню на рис. 2.8 перечислены *дистрибутивные наборы*. Режим `Custom` дает возможность выбирать отдельные дистрибутивы.)

Среда инсталляции

FreeBSD может инсталлироваться из самых разных источников.

- **CD/DVD**. При выборе опции `CD/DVD` система будет инсталлироваться с локального дисковода CD-ROM или DVD-ROM. Если есть дистрибутивный диск, то это самый быстрый и удобный метод.
- **FTP**. Существуют три варианта FTP-инсталляции: `FTP` (обычный режим), `FTP Passive` (пассивный режим, допускаемый некоторыми брандмауэрами, которые блокируют обычные FTP-запросы) и `HTTP` (взаимодействие с FTP-сервером через прокси-сервер поверх протокола HTTP). Все эти варианты позволяют инсталлировать FreeBSD с одного из нескольких официальных FTP-узлов или с локального FTP-сервера (в этом случае нужно в списке FTP-узлов выбрать пункт `URL`, а затем ввести адрес FTP-сервера и название каталога, где расположены файлы FreeBSD). FTP-инсталляция подойдет лишь в том случае, когда есть высокоскоростное сетевое соединение и нет дистрибутивного компакт-диска.
- **NFS**. *Сетевая файловая система* (Network File System, NFS) широко используется в UNIX для организации совместного доступа к файлам. Исходные файлы FreeBSD можно поместить на локальный сервер NFS, с которого их можно будет инсталлировать, как с FTP-сервера.
- **Локальный жесткий диск**. Опции `DOS` и `File System` означают инсталляцию системы с локального жесткого диска. В первом случае нужно выбрать дисковый раздел из предлагаемого списка, во втором — задать путь к смонтированной файловой системе. Второй вариант используется при обновлении системы.
- **Дискета или магнитная лента**. Две оставшиеся опции — `Floppy` и `Tape`. В настоящее время дискеты и магнитные ленты крайне редко используются в качестве инсталляционных носителей. FreeBSD занимает слишком много дискет, а магнитные ленты гораздо менее удобны, чем компакт-диски.

Процедура инсталляции практически не зависит от того, где находятся исходные файлы. Если используется съемный носитель, программа попросит вставить его в дисковод. В случае сетевой инсталляции параметры конфигурации сети нужно ввести раньше, чем при инсталляции с компакт-диска.

Непосредственная установка системы

После выбора способа инсталляции программа спросит можно ли приступать к установке системы Будет выдано предупреждение о том что это последний шанс внести изменения или прервать инсталляцию Получив подтверждение, инсталлятор создаст файловые системы в разделах FreeBSD и копирует туда необходимые файлы

В ходе инсталляции программа информирует пользователя о выполняемых действиях. Продолжительность инсталляции зависит от скорости жесткого диска и от того где находятся исходные файлы (на компакт-диске или на сервере), а также от объема инсталлируемого программного обеспечения.

Начальное конфигурирование системы

Создав разделы FreeBSD и скопировав в них файлы, программа `sysinstall` приступит к настройке основных параметров системы. Важнее всего правильно задать параметры сети, сконфигурировать пакет XFree86 (если он инсталлирован) и создать пользовательские учетные записи

Настройка параметров сети

Инсталлятор спросит нужно ли задать параметры протоколов Ethernet или SLIP/PPP. Если это не требуется, выберите `No`, в противном случае — `Yes`. Ниже предполагается, что задаются параметры для сети Ethernet. Детали настройки протоколов SLIP (используется при соединении компьютеров последовательными кабелями) и PPP (используется при подключении компьютера к Internet через модем) будут отличаться.

Прежде всего следует задать тип сетевого интерфейса (рис. 2.9). Если в компьютере установлена плата Ethernet, то нужно выбрать первый элемент списка. Интерфейс `sl0` соответствует протоколу SLIP, а интерфейс `ppp0` — протоколу PPP. Учтите, что конкретное название интерфейса зависит от набора микросхем Ethernet, установленных на плате. Во многих системах будут присутствовать дополнительные опции соответствующие последовательным и параллельным портам

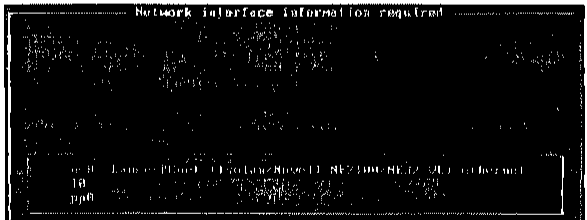


Рис. 2.9. FreeBSD автоматически обнаруживает сетевую плату Ethernet и позволяет выбрать тип сетевого соединения

Далее программа спросит, нужно ли включить поддержку IPv6 — сетевого протокола следующего поколения, который пока не нашел широкого применения Если

выбрать No, программа предложит настроить DHCP (Dynamic Host Configuration Protocol — протокол динамического конфигурирования компьютеров) Этот протокол позволяет серверу назначать IP-адреса и другие сетевые параметры клиентским системам. Его применение способно значительно упростить сетевое конфигурирование Если ответить Yes, программа свяжется с DHCP-сервером и отобразит полученные от него значения (рис. 2.10). В противном случае придется вводить эти значения вручную (проконсультируйтесь с сетевым администратором). Особенно важны IP адрес сетевая маска, адрес шлюза и адрес сервера DNS. Формат адреса — 4-байтовое число, компоненты которого разделены точкой например 192.168.1.27 Сетевой администратор может предоставить адреса нескольких серверов DNS. Выберите один (в главе 14 будет показано, как задать насколько серверов в системе, которая уже функционирует). С технической точки зрения имя компьютера и доменное имя не обязательны для нормального функционирования системы но они широко применяются поэтому лучше все же ввести их

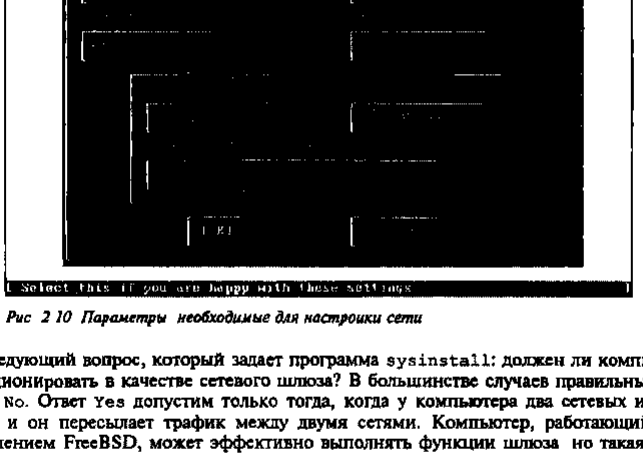


Рис 2 10 Параметры необходимые для настройки сети

Следующий вопрос, который задает программа sysinstall: должен ли компьютер функционировать в качестве сетевого шлюза? В большинстве случаев правильный ответ — No. Ответ Yes допустим только тогда, когда у компьютера два сетевых интерфейса и он пересылает трафик между двумя сетями. Компьютер, работающий под управлением FreeBSD, может эффективно выполнять функции шлюза но такая конфигурация выходит за рамки книги.

Далее система предложит сконфигурировать демон inetd. Это так называемый *суперсервер* — программа контролирующая попытки подключения к другим серверным программам компьютера и соответствующим образом координирующая трафик. Не все серверные программы работают под управлением демона inetd Поддерживают его, в частности серверы Telnet и FTP Если ответить Yes, программа sysinstall предупредит о том, что серверы, контролируемые демоном inetd, повышают уязвимость системы, и предложит передумать. При повторном ответе Yes программа запустит редактор в котором можно изменить конфигурационный файл демона — /etc/inetd.conf. Найдите в этом файле строки, соответствующие запускаемым серверам, и удалите начальные символы комментарив (#) По окончании редактирования

нажмите <Ctrl+[>, выберите в появившемся меню опцию a (`leave editor`) и снова нажмите a (`save changes`)

Теперь программа задаст ряд вопросов касающихся конфигурации конкретных серверов

- **Анонимный FTP-сервер** Программа спросит нужно ли разрешить анонимный доступ к FTP-серверу Если не знаете, что это означает, ответьте `No`. Поддержку анонимного доступа *не нужно* включать для работы с другими FTP серверами Она необходима только для локального анонимного FTP-сервера
- **Сервер NFS**. Сервер NFS позволяет клиентам FreeBSD и других UNIX-систем получать доступ к файлам компьютера, как если бы это были их локальные файлы Если не уверены в необходимости этого, ответьте `No` В случае положительного ответа придется отредактировать конфигурационный файл сервера NFS `/etc/exports` (глава 18)
- **Клиент NFS** Если ответить `Yes` на запрос по поводу NFS клиента система разрешит монтировать NFS-каталоги других UNIX-систем.
- **Профиль безопасности**. FreeBSD располагает множеством средств защиты Ответив `Yes` на запрос по поводу профиля безопасности, можно задать значения целого ряда опций Если ответить `No`, будет выбран профиль `Medium`.

Разрешается вообще не задавать никаких сетевых параметров на данном этапе, отключив все сетевые протоколы. Безопаснее оставлять сетевые опции отключенными потому что каждая серверная программа это потенциальная "брешь" в защите системы Сетевое программное обеспечение можно сконфигурировать когда в этом возникнет необходимость

Установки системной консоли

Если планируется часто работать в текстовом режиме имеет смысл настроить параметры консоли

- **Шрифт**. Стандартный шрифт подходит для большинства случаев, но можно выбирать альтернативные наборы символов, которые в основном предназначены для отображения алфавитов, отличных от латиницы
- **Раскладка клавиатуры** Можно заставить FreeBSD использовать например французскую или русскую раскладку
- **Скорость повтора**. Можно выбрать одну из четырех скоростей ввода повторных символов при удержании клавиши К сожалению, в процессе инсталляции нельзя проверить выбранную установку
- **Экранная заставка**. Опция `Saver` позволяет установить одну из экранных заставок текстового режима Заставка появится на экране, если пользователь в течение заданного времени не будет работать за компьютером Для сбережения электроэнергии особенно полезна заставка `Green` При ее активизации современные мониторы переходят в режим пониженного энергопотребления
- **Таблица шрифтов** Можно выбрать несколько методов преобразования UNIX символов в экранные шрифты Это необходимо, когда используется шрифт отличный от латиницы
- **Эмуляция терминала** Система умеет эмулировать различные типы терминалов используемые программами для адресации экрана

Часовой пояс

Настройка часового пояса очень важна, особенно для UNIX-систем Пропускать этот этап нежелательно. Сначала задается вопрос, должен ли системный таймер ориентироваться на UTC (Universal Time Coordinated — всеобщее скоординированное время). Традиционно в UNIX значения времени хранились в формате UTC и преобразовывались в местное время динамически, в соответствии с часовым поясом Но в системах на базе процессоров x86 все происходит наоборот: значения времени хранится в местном формате и преобразуются в формат UTC для тех протоколов, которым это необходимо. Выберите тот метод преобразования, который считаете нужным

Совет

Если на компьютере установлена только FreeBSD либо FreeBSD и другая UNIX-система, например Solaris или Linux, заставьте системный таймер работать в формате UTC. Это снизит риск возникновения проблем при переходе на летнее время Если же FreeBSD установлена совместно с DOS, Windows или OS/2, выберите местный формат времени При этом каждую весну и осень нужно будет повторно инициализировать системный таймер, зато перезагрузка в другой операционной системе не приведет к сбою часов.

Часовой пояс устанавливается в три этапа Сначала нужно выбрать географический регион, например `America`, затем — страну в рамках региона, например `United States`. Наконец, выбирается часовой пояс в пределах страны (этот этап не всегда необходим) После этого система сообщает трехсимвольный код часового пояса, такой как `EST`, и запрашивает подтверждение Если вы точно знаете, что это неправильный код, выберите `No` и поменяйте установки

Совместимость с исполняемыми файлвми Linux

Во FreeBSD могут выполняться программы, скомпилированные в Linux Несмотря на то что для большинства Linux программ существуют эквиваленты во FreeBSD, режим бинарной совместимости бывает удобен, когда соответствующие исполняемые файлы недоступны во FreeBSD, а исходный код либо отсутствует, либо плохо компилируется

Linux и FreeBSD настолько похожи, что для запуска Linux-программ во FreeBSD требуется поддержка лишь нескольких библиотек и компонентов ядра. По этой причине обычно рекомендуется включать режим бинарной совместимости с Linux В ответ программа `sysinstall` загрузит соответствующий модуль ядра Подробнее о применении режима совместимости рассказывается в главе 4

Настройка параметров мыши

FreeBSD поддерживает текстовый указатель мыши, позволяющий вырезать и вставлять фрагменты в текстовом режиме. В первую очередь программа `sysinstall` спрашивает, подключена ли к системе USB-мышь Если ответить `No`, появится меню настройки и тестирования мыши (рис. 2.11).

- **Тип мыши** При выборе команды `Type` можно задать программный протокол используемый мышью Современные мыши в основном работают по протоколу PS/2 (тип `Auto`) FreeBSD позволяет также выбрать один из протоколов для старых устройств, подключаемых к последовательному порту RS-232 `Logitech` и `Microsoft`

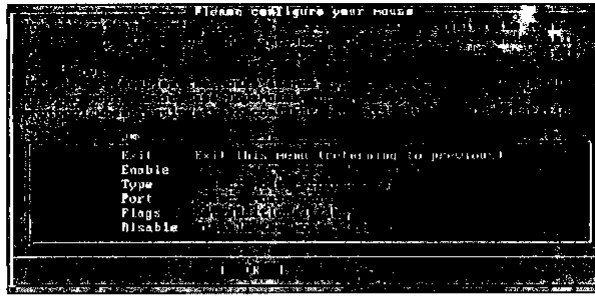


Рис 2 11 Меню для задания параметров мыши

- **Порт мыши** Команда `Port` предназначена для задания порта, к которому подключена мышь Возможные варианты таковы `PS/2`, `COM1`—`COM4` (последовательный порт RS 232) и `BusMouse` (старые устройства с шинным интерфейсом) Учтите, что протокол PS/2 и аппаратный порт PS/2 – разные понятия
- **Дополнительные параметры.** Те, кто хорошо знакомы с работой демона мыши могут задать дополнительные параметры воспользовавшись командой `Flags` Необходимость в этом возникает редко
- **Тестирование и включение** Выберите команду `Enable`, чтобы протестировать демон мыши. На экране должен появиться указатель мыши, реагирующий на ее перемещения (кнопки не должны работать) Ответьте на вопрос, работает ли устройство. Если это не так, попробуйте другие установки

Если мышь по какой-то причине не работает, отключите демон мыши, выбрав команду `Disable` Его можно включить позднее, при настройке пакета XFree86 или по окончании инсталляции системы А пока что с системой можно работать в текстовом режиме

По завершении конфигурирования мыши выберите команду `Exit` для перехода к следующему этапу Система спросит, нужно ли настраивать X-сервер Удобнее сделать это сейчас, а не после инсталляции, но иногда в процессе конфигурирования сервера возникают проблемы, которые приводят к зависанию системы Если в компьютере установлен какой-нибудь экзотический видеоадаптер, можно пропустить этап конфигурирования X-сервера и вернуться к нему позднее, запустив программу `sysinstall` в работающей системе

Конфигурирование пакета XFree86

В главном конфигурационном меню пакета XFree86 содержатся четыре команды (не считая стандартной команды `Exit`)

- **`xf86cfg`** Это графическая утилита конфигурирования X-сервера Она корректно работает только в том случае, когда графическая плата поддерживает стандартные драйверы VGA Оптимальные драйверы можно будет установить по завершении инсталляции системы
- **`xf86cfg -textmode`** То же, что и предыдущее, но утилита `xf86cfg` запускается в текстовом режиме, благодаря чему систему можно конфигурировать, даже если графическая плата не поддерживает драйвер VGA

- **xf86config** Это еще одна текстовая утилита конфигурирования X сервера, не столь гибкая как `xf86cfg` Она задает ряд вопросов, касающихся графической платы, мыши и др. Если допустить ошибку, придется перезапускать программу. Пользоваться ею рекомендуется лишь в крайнем случае.
- **XDesktop** Эта команда конфигурирует не X-сервер а лишь рабочий стол X-среды. Она сообщает системе установки по умолчанию, касающиеся типов окон, значков рабочего стола и т.д Применять ее имеет смысл в том случае когда в системе уже сконфигурирован X-сервер и нужно модифицировать лишь параметры рабочего стола.

Большинство пользователей предпочитают первый вариант Если предварительно была сконфигурирована мышь, программа `sysinstall` сообщит установки, которых следует придерживаться. Выберите OK, и система попросит нажать <Enter> для переключения в графический режим. Экран очистится, и на сером фоне появится крестообразный курсор, а затем — главное окно утилиты `xf86cfg` (рис 2 12) со сводкой конфигурационных настроек Чтобы сконфигурировать X-сервер, активизируйте по очереди каждый значок в верхней части окна. Для этого наведите на него указатель щелкните правой кнопкой мыши и выберите в раскрывшемся меню команду Configure. Появится окно с перечнем параметров соответствующего устройства. Каждому устройству нужно назначить имя. Стандартное имя определяется типом устройства например `Keyboard1` для клавиатуры, и редко требует изменения

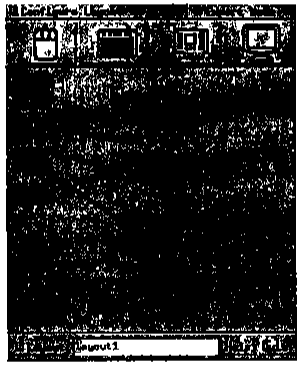


Рис 2.12. Утилита `xf86cfg` запускает простейший X-сервер VGA, позволяющий настраивать тот сервер, который будет работать во FreeBSD

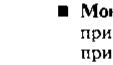
Совет Если при вызове утилиты `xf86cfg` экран очищается и указатель мыши остается неподвижным, нажмите <Ctrl+Alt+Backspace> Это приведет к завершению работы X-сервера, и можно будет сделать повторную попытку На этот раз воспользуйтесь командой `xf86cfg-textmode`, которая аналогична `xf86cfg`, но работает в текстовом режиме

Ниже перечислены устройства требующие конфигурирования

- **Мышь**. Если мышь была активизирована ранее, выберите из списка протоколов SysMouse и введите **/dev/sysmouse** в поле Select Mouse Device (демон мыши взаимодействует непосредственно с аппаратным устройством и записывает получаемые от него данные в псевдоустройство /dev/sysmouse, где они доступны X-серверу). В противном случае задайте соответствующий протокол и физическое устройство Для мыши PS/2 это должно быть устройство /dev/psm0, для мыши USB — устройство /dev/ums0, для последовательных портов RS-232 (COM1 -COM4 в DOS и Windows) — устройства /dev/cuaa0— /dev/cuaa3. Всем устройствам PS/2 и USB соответствует протокол PS/2. Если используется двухкнопочная мышь, включите режим эмуляции третьей кнопки (чтобы имитировать ее нажатие, нужно нажать обе кнопки одновременно).

Многие X-программы предполагают наличие 3-кнопочной мыши. Если у имеющейся мыши всего две кнопки, лучше заменить ее 3-кнопочной моделью, так как режим эмуляции неудобен

- **Клавиатура**. Утилита позволяет задавать тип клавиатуры В США используются 101- и 104-клавишная модели (на последних в нижнем ряду есть кнопки с логотипом Windows). Неамериканские клавиатуры имеют несколько иную раскладку. При выборе клавиатуры отображается ее схема которую можно сравнить с имеющимся образцом
- **Видеоадаптер** Чрезвычайно важно правильно сконфигурировать графическую плату Если модель платы присутствует в предлагаемом списке, выберите ее Тем самым будет определен видеодрайвер, используемый пакетом XFree86 Вместо названия платы в списке может присутствовать название набора микросхем Если же и его нет, задайте универсальный графический режим, щелкнув на кнопке Driver и выбрав опцию VGA или VESA. Для некоторых плат подойдет устройство fbdev, применение которого обычно вызывает снижение производительности Неправильный выбор приведет к тому, что X-сервер не сможет запуститься
- **Монитор** Утилита отображает список распространенных типов мониторов, например "High Frequency SVGA, 1024x768 @ 70 Hz". Если имеющийся монитор присутствует в списке, выберите его Узнайте в руководстве к монитору, каковы диапазоны горизонтальной и вертикальной частот развертки, и задайте эти значения в соответствующих полях. Это позволит X-серверу максимально эффективно использовать возможности монитора.



Старые мониторы могут быть повреждены при выборе слишком высокого разрешения или слишком большой частоты обновления Соблюдайте осторожность при задании параметров монитора Не выбирайте тип монитора, который поддерживает завышенные значения разрешения или частоты обновления К счастью, мониторы, выпускаемые со второй половины 90-х, игнорируют сигналы, выходящие за установленные рамки, поэтому их нельзя повредить таким способом При выборе некорректного режима изображение на мониторе пропадет или появится сообщение о том, что монитор не поддерживает данный режим

- **Экран** В терминологии X Window System *экраном* называется конкретное сочетание установок для графической платы и монитора В верхней части окна утилиты xf86cfg находится меню Configure Layout (см рис 2 12) Щелкните на нем и выберите команду Configure Screen Появится большое изображение монитора Щелкните на нем правой кнопкой мыши и выберите Configure В

правой части окна будет приведен список разрешений (с указанием цветовой разрядности), которые X-сервер попытается использовать. Минимальный размер экрана, с которым имеет смысл работать, — 800×600. На это разрешение по умолчанию настроены многие программы. Более удобный вариант — 1024×768. Чем больше разрешение, тем лучше, при условии, что видеоплата и монитор его поддерживают.

По окончании настройки конфигурационных параметров щелкните на кнопке Quit в нижней части окна и протестируйте установки. После этого X-сервер завершит свою работу.

Конфигурирование рабочего стола X-среды

Далее программа `sysinstall` предложит сконфигурировать рабочий стол. На выбор предлагается несколько графических оболочек:

- **KDE**. *K Desktop Environment* (KDE) — это одна из двух основных встроенных графических оболочек FreeBSD. Если система поставляется на одном компакт-диске, то большинство файлов KDE отсутствует, так что этот вариант может быть недоступен.
- **GNOME+Sawfish или GNOME+Enlightenment**. GNOME (GNU Network Object Model Environment) — другая встроенная графическая оболочка FreeBSD. В дополнение к ней могут использоваться два *диспетчера окон* — Sawfish и Enlightenment (диспетчер окон контролирует расположение окон, их перемещение и т.д.). Программа Sawfish менее требовательна к ресурсам, чем Enlightenment, и потому является предпочтительным выбором в системах с не большим объемом памяти, формируя более быстродействующую настольную среду. Программа Enlightenment обладает более гибкими средствами настройки рабочего стола.
- **Afterstep**. Это диспетчер окон, смоделированный по образцу оконной среды старой операционной системы NeXT.
- **Windowmaker**. Этот диспетчер окон заимствовал концепции многих программ. Он составляет ядро пакета GNUStep (`http / gnustep.org`), который позволяет реализовать спецификацию OpenStep, используемую в операционной системе NeXT. При выборе программы Windowmaker сам пакет GNUStep не инсталлируется. Его можно добавить позднее.
- **fvwm**. Программа `fvwm` является самым старым из традиционных диспетчеров окон.

Последние три варианта — это диспетчеры окон. Они нетребовательны к ресурсам памяти и хорошо подходят для рабочих станций. KDE и GNOME — это огромные графические оболочки, занимающие несколько мегабайт оперативной памяти. Зато они формируют полноценную настольную среду, располагая диспетчерами файлов, утилитами конфигурирования и множеством других средств. Любая из этих оболочек вполне подойдет для пользователей, привыкших работать в Windows или Mac OS.

Важно понимать, что это лишь *стандартные* оболочки. Можно инсталлировать сколько угодно диспетчеров окон и графических оболочек. Пользователи могут настроить свои учетные записи на запуск любой из имеющихся оболочек, даже если она отличается от той, что установлена по умолчанию. Подробнее об этом рассказывается в главе 13.

Следует также отметить, что многие утилиты, включая компоненты оболочек GNOME и KDE, могут работать в любой оболочке, хотя некоторые программы лучше функционируют в "однородной" среде. Например, можно задать стандартный набор

шрифтов, применяемых в текущей графической оболочка и все написанные для нее программы будут работать с этим набором шрифтов. Программы, написанные для другой оболочки, не всегда распознают эти установки.

Какая графическая оболочка лучше? Это зависит от личных предпочтений Если в системе достаточно памяти (не менее 128 Мбайт), установите GNOME или KDE. Если необходима менее ресурсоемкая среда, установите только диспетчер окон Конкретный выбор здесь тоже определяется личными предпочтениями

Конфигурирование программных пакетов

По завершении конфигурирования X-среды программа `sysinstall` предложит просмотреть коллекцию пакетов FreeBSD Пакеты разбиты на категории, такие как `afterstep` или `audio` В данный момент инсталлировать их необязательно, это можно сделать и позднее. Выберите лишь те программы которые точно будут использоваться

Совет

Важные программы находятся в категории `editors`. По умолчанию FreeBSD инсталлирует лишь несколько простейших редакторов, в частности `ee` и `vi`. Первый из них используется для редактирования конфигурационных файлов (например, `/etc/inetd.conf`) в ходе инсталляции системы. Редактор `vi` имеет определенный круг почитателей, хотя поначалу с ним довольно непривычно работать. Чрезвычайно популярен редактор `emacs`, но он сложен для изучения. Его разновидностями являются редакторы `uemacs`, `jed` и `pico`. Существуют также два графических редактора. `gedit` и `nedit`. Не все они доступны, когда система поставляется на одном компакт-диске,

Нвстройка учетных звписей

После установки дополнительных программных пакетов программа `sysinstall` позволит создать учетные записи пользователей. Прежде чем переходить к деталям следует поговорить о том, зачем нужны учетные записи и как их конфигурировать

Значение учетных записей

Старые системы на базе процессоров *x*86, такие как DOS и Windows 9*x*/Me являются преимущественно однопользовательскими Предполагается, что за компьютером работает один человек или, по крайней мере, небольшая группа людей, доверяющих друг другу (например, члены семьи). Каждый пользователь может просматривать и модифицировать любые файлы на компьютере, включая личные файлы других пользователей, а также системные файлы, в частности шрифты драйверы, программные файлы и т д.

На заметку

Можно заставить Windows 9*x*/Me и даже DOS отображать экран регистрации с требованием ввести имя пользователя и пароль. В сетевой среде таким способом ограничивается доступ пользователей к некоторым сетевым ресурсам, но содержимое локального компьютера будет по-прежнему открыто для erb пользователей Окно регистрации в Windows 9*x*/Me содержит кнопку Cancel, позволяющую кому угодно войти в систему без доступа к сети не вводя ни имя пользователя ни пароль

В противоположность этому UNIX-системы давным-давно поддерживали много пользовательские операции. В 70-е и 80-е годы UNIX-системами были в основном мини-компьютеры и большие ЭВМ с которыми работали через удаленные термина лы, и в системе одновременно разрешалось регистрироваться множеству пользовате лей. Разработчики и администраторы UNIX не могли доверять пользователям до та кой степени, как в среде персональных компьютеров. Поэтому в UNIX издавна суще ствовали средства, позволявшие ограничить доступ пользователей к ресурсам друг друга и самой системы FreeBSD унаследовала эти средства, основанные на концеп ции *учетных записей*

Учетная запись представляет собои набор структур данных, создаваемых операци онной системой для отдельного пользователя или для определенного применения от имени "псевдопользователя" С каждой учетной записью связаны следующие элементы

- **Имя пользователя** Это имя, закрепленное за учетной записью, например `bjol_tz` Пользователь вводит свое имя, чтобы получить доступ в систему. Имена пользователей используются и в других целях, в частности при рассыл ке электронной почты
- **Пароль** *Пароли* — основное средство защиты FreeBSD Теме паролей будет по священ отдельный раздел в главе 29, в которой обсуждаются вопросы безопасно сти Помните о том, что в качестве пароля не следует выбирать слова или имена. Идеальный пароль состоит из произвольного сочетания букв, цифр и знаков пунктуации Пользователь обязан ввести пароль при регистрации в системе.
- **Группа** В UNIX применяется концепция групп пользователей. *Группа* — это несколько пользователей, имеющих доступ к конкретным файлам или про граммам Например, можно создать группу, членам которой разрешен доступ к файлам совместного проекта
- **Идентификаторы пользователя (UID) и группы (GID).** Значения UID и GID — это числа, идентифицирующие имя пользователя и название группы Они применяются самой системой, которая автоматически преобразует их в тексто вый вид при взаимодействии с пользователями. Например, имени пользовате ля `bjo__tz` может соответствовать идентификатор 1098.
- **Начальный каталог** В большинстве систем резервируется место на диске, где пользователи могут хранить свои файлы. Чтобы избежать путаницы и сохра нить четкую структуру каталогов, каждому пользователю выделяется собствен ный *начальный каталог* По умолчанию начальные каталоги располагаются в разделе `/home` и называются в соответствии с именем владельца, например `/home/bjol.tz` В главе 10 будет показано как изменить начальный каталог пользователя
- **Интерпретатор команд** Это программа, принимающая от пользователя команды и обрабатывающая их FreeBSD поддерживает несколько интерпретаторов, ра ботающих в режиме командной строки, в частности `tcsh` `bash` и `sh` (используется по умолчанию) Работой в графической среде управляет диспет чер окон или графическая оболочка Соответствующая установка задается в конфигурационных файлах, хранящихся в начальном каталоге пользователя.

Кроме учетных записей обычных пользователей во FreeBSD есть несколько специ альных учетных записей, применяемых системой для внутренних целей. Среди них — `root`, `daemon`, `b_r` и `nobody` Некоторым программам нужны отдельные учетные за писи, которые требуется создавать при инсталляции программы Все эти *системные учетные записи* имеют идентификаторы от 0 до 100 Идентификаторы учетных записей обычных пользователеи — от 1000 и выше. Существует также несколько стандартных системных групп `root` `wheel` и `nobody`

Не все UNIX системы назначают обычным пользователям идентификаторы начиная от 1000 К примеру, в некоторых дистрибутивах Linux эти идентификаторы отсчитываются от 500 и даже от 100. Поскольку идентификаторы нужны операционной системе для распознавания пользователей указанные различия могут оказаться существенными в некоторых ситуациях. Допустим, когда сервер NFS предоставляет пользователю доступ к своим файлам, он проверяет значение UID Если оно не совпадает на сервере и клиенте, пользователю может быть отказано в доступе В сети с небольшим числом пользователей следует назначать пользователям одинаковые идентификаторы на всех компьютерах.

Некоторым системным учетным записям разрешен привилегированный доступ к определенным файлам или другим системным ресурсам. Возможности такой учетной записи, как nobody, наоборот, сильно ограничены. FreeBSD использует эти различия в целях обеспечения безопасности чтобы предоставить некоторым процессам доступ к файлам и устройствам, которые в обычном режиме недоступны.

Особенно важной является учетная запись root Ее идентификаторы пользователя и группы равны нулю и этой учетной записи разрешен доступ ко всей системе. Системный администратор (называемый *суперпользователем*) применяет учетную запись root в целях обслуживания системы Лишь очень немногие из числа административного персонала должны иметь доступ к этой учетной записи, а пароль суперпользователя должен быть надежно защищен Человек, инсталлирующий FreeBSD, обычно является суперпользователем, хотя эту привилегию можно впоследствии передать кому то другому

У суперпользователя как правило, имеется и обычная учетная запись Именно она используется для повседневной работы — обработки почты просмотра Web-страниц, компиляции программ и т д Когда возникает необходимость выполнить действия от имени суперпользователя на помощь приходит команда su, описываемая в главе 3

Не используйте учетную запись root для выполнения повседневных задач, таких как работа в Internet или редактирование текста Поскольку суперпользователь имеет полный доступ ко всей системе, можно нечаянно нанести серьезный вред системе. Например, достаточно одной опечатки при вводе команды rm, чтобы удалить все файлы компьютера FreeBSD не имеет средств защиты от подобных оплошностей Чтобы избежать катастроф назначайте пользователям ограниченные права доступа

Получив базовую информацию, продолжим рассматривать процесс инсталляции FreeBSD На данном этапе программа sysinstall предлагает создать учетные записи

Создание группы

В ходе инсталляции можно создавать учетные записи как пользователей, так и групп Если заранее известно, как будут сгруппированы пользователи, начните с создания групп, воспользовавшись командой Group в меню User and Group Management. Появится окно, в котором нужно ввести название группы, задать ее идентификатор и выбрать ее членов (это поле можно оставить пустым, если учетные записи пользователей отсутствуют). Желательно использовать идентификаторы от 1000 и выше

В некоторых разновидностях UNIX распространена такая конфигурация: все учетные записи обычных пользователей включаются в группу users. Это удобный подход, когда в системе всего несколько пользователей и нужно предоставить им одинаковый доступ к определенным файлам и каталогам Другой часто применяемый подход

создать для каждого пользователя отдельную *приватную* группу. В такую группу можно впоследствии добавлять других пользователей, разрешая им доступ к файлам конкретного пользователя.

На заметку

Пользователи могут входить сразу в несколько групп. Таким образом группу `users` можно дополнять другими группами, ограничивая доступ к определенным файлам и каталогам.

Если в той или иной группе нет особой необходимости, то не нужно и создавать ее. Для начала достаточно создать группу `users`, сделав ее группой по умолчанию для всех новых пользователей.

Создание учетной записи

При выборе команды `User` в меню `User and Group Management` появится окно, представленное на рис. 2.13. Большинство его полей следует заполнить (в некоторые поля записываются значения по умолчанию).

Рис. 2.13. В процессе инсталляции операционной системы разрешается создавать различные учетные записи

В поле `Login ID` задается имя пользователя, которое должно быть уникальным на данном компьютере (если пользователь располагает учетными записями на разных компьютерах, то по возможности ему следует назначать одинаковые имена во избежание путаницы). Во FreeBSD имена пользователей чувствительны к регистру символов, при этом большинство имен представлено в нижнем регистре. В поле `Full Name` можно внести произвольную информацию о пользователе. Обычно здесь указываются его имя и фамилия.

Поле `Group` содержит название начальной группы, в которую входит пользователь. Эта группа будет назначаться файлам, создаваемым этим пользователем, и программам, которые он запускает. В поле `Member groups` указываются названия других групп, к которым принадлежит пользователь. Пользователь сможет получать доступ к файлам этих групп, а создаваемые им в каталоге одной из этих групп файлы будут доступны членам группы. Поле `Member groups` разрешается оставлять пустым.

Совет Одна из специальных групп, создаваемых системой, называется `wheel`. Ее члены имеют право пользоваться командой `su` для получения привилегий других пользователей, включая `root`. Только администраторы, которым разрешен привилегированный доступ к компьютеру, должны быть членами группы `wheel` (она может задаваться в поле `Group` либо `Member groups`).

Значения, отображаемые в полях `Home directory` (начальный каталог) и `Login shell` (регистрационный интерпретатор команд), желательно оставить в том виде, в каком они предлагаются по умолчанию.

На заметку Любители интерпретатора `bash` должны учитывать, что он не инсталлируется по умолчанию. Прежде чем указывать его в качестве атрибута учетной записи, позаботьтесь об установке соответствующего пакета. Интерпретатор `bash` инсталлируется в каталог `/usr/local/bin`, а не `/bin`, где находится большинство интерпретаторов. Если попытаться задать несуществующий интерпретатор, программа `sysinstall` сообщит об этом.

При инсталляции системы создайте учетную запись только для себя. Остальные учетные записи можно создать позже с помощью программы `sysinstall` или специальных утилит наподобие `adduser`.

Завершение инсталляции

После создания всех учетных записей воспользуйтесь командой `Exit`. Далее программа попросит задать пароль суперпользователя, после чего предложит перейти в меню конфигурирования системы для повторной настройки некоторых параметров. Затем программа вернется в главное меню (см. рис. 2.5). Выберите `Exit Install`, чтобы завершить инсталляцию. Программа сообщит о том, что она собирается перезагрузить компьютер. Удалите из дисковода загрузочную дискету или компакт-диск и дождитесь перезагрузки.

В процессе перезагрузки появится приглашение системного загрузчика. Если при инсталляции загрузчика были выбраны установки по умолчанию и на компьютере установлена DOS или Windows, на экране будут присутствовать строки вида `F1 DOS` и `F2 FreeBSD`. Соответственно, нужно нажать <F1>, чтобы загрузиться в DOS, и <F2> для загрузки во FreeBSD. Сделанный выбор становится установкой по умолчанию для следующей перезагрузки. Если на компьютере установлена только FreeBSD, то других вариантов не будет. Вместо меню загрузчика FreeBSD может отображаться меню стороннего загрузчика, если таковой установлен. Подробнее о настройке параметров загрузчика рассказывается в главе 4.

Резюме

Процесс инсталляции FreeBSD включает в себя получение исходных файлов системы, подготовку компьютера и запуск инсталлятора. При установке системы нужно указать, куда следует установить FreeBSD, как разбить жесткий диск на разделы и какие компоненты системы инсталлировать. Попутно можно сконфигурировать различные устройства (мышь, графическую плату и т.д.), а также серверное программное обеспечение. Некоторые аспекты инсталляции сложны для новичков, но значения по умолчанию вполне разумны. Впоследствии можно поменять конфигурацию системы или даже полностью переинсталлировать ее, если возникнет такая необходимость. Большая часть последующих глав посвящена именно настройке системы.

Полный справочник по

Глава 3

Обзор системы

Итак, FreeBSD установлена на компьютере, и можно переходить к знакомству с системой. В этой главе дан краткий обзор FreeBSD: мы познакомимся с процедурами начальной загрузки, регистрации в системе, запуска программ, работы в графической среде и останова системы. Если FreeBSD установлена на настольном компьютере, все эти действия придется выполнять достаточно регулярно. Но есть системы, которые работают без перезагрузки днями и даже месяцами, есть серверы, где не требуется наличие графической среды, поэтому некоторые задачи будут менее актуальными

Начальная загрузка

В определенном смысле загрузка компьютера работающего под управлением FreeBSD, — очень простой процесс Нужно лишь включить питание и подождать, пока проинициализируется операционная система. А "за кулисами", тем временем происходит множество событий, о которых полезно знать особенно если необходимо устранить неполадку

Выбор FreeBSD

На компьютерах архитектуры *x86* начальной загрузкой управляет BIOS (Basic Input/Output System — базовая система ввода-вывода) После завершения тестов памяти и других подготовительных действий, BIOS передает управление начальному загрузчику который обычно находится на жестком диске (реже — дискете компакт диске или другом съемном носителе)

Если на компьютере инсталлирована не только FreeBSD, но и другая операционная система, появится приглашение начального загрузчика примерно следующего вида (приглашение *этапа 0*)

```
F1    DOS
F2    FreeBSD

Default: F2
```

FreeBSD помечает разделы Windows меткой *DOS* так что не удивляйтесь подобному обозначению. Нажмите клавишу, соответствующую выбираемой операционной системе (<F1> <F2> или другую, если вариантов для выбора больше двух) Значение по умолчанию соответствует операционной системе, загружавшейся в прошлый раз Оно будет выбрано автоматически через некоторое время

На заметку На компьютере может быть установлен не загрузчик FreeBSD а другой загрузчик с соответствующим интерфейсом Об этом рассказывается в главе 4

Когда начинается загрузка системы, на экране появляются статусные сообщения Первое из них выдается загрузчиком системы который является вторичным загрузчиком (в терминологии FreeBSD он известен как загрузчик *этапов 1 и 2*; в действительности это две разные программы но они функционируют как одно целое) Начальный загрузчик этапа 0 представляет собой очень маленькую и простую программу. Он просто направляет процесс загрузки в нужный раздел. Загрузчик этапов 1–2 более сложен и специфичен для FreeBSD. Он отображает примерно следующее сообщение:

```
Hit [Enter] to boot immediately, or any other key for command prompt
Booting [/boot/kernel/kernel] in 9 seconds..
```

Счетчик во второй строке уменьшается до нуля. Можно подождать или сразу нажать <Enter>, чтобы немного ускорить процесс. В редких случаях требуется загрузить специальные модули ядра (аппаратные драйверы) или как-то иначе сконфигурировать ядро. Если это так, нажмите любую другую клавишу, и загрузчик системы позволит ввести необходимые команды. Некоторые из них описаны в главе 4.

Далее загрузчик записывает в память ядро FreeBSD, хранящееся в файле `/boot/kernel/kernel` (по умолчанию). Ядро отображает статусную информацию и приступает к вызову сценариев запуска системы.

Интерпретация сообщений ядра

В отличие от Windows, которая выводит не так много загрузочных сообщений, FreeBSD довольно "многословна" в процессе загрузки. Количество отображаемых сообщений столь велико, что они "проносятся" по экрану прежде, чем их можно прочесть. К счастью, сообщения ядра хранятся некоторое время по завершении загрузки и их можно снова просмотреть, воспользовавшись командой `dmesg`. Проще всего перенаправить их в файл с помощью команды вида `dmesg > dmesg.txt` или переслать программе постраничной разбивки (например, `dmesg | less`). Это, впрочем, можно сделать только после регистрации в системе, о чем пойдет речь ниже.

На заметку Перенаправление потоков ввода-вывода в файлы и механизм каналов являются распространенными методами организации межзадачного взаимодействия во FreeBSD. Подробнее об этом рассказывается в главе 5.

Сообщения ядра — важный источник информации о системе. Сначала выводятся сообщения об авторских правах и сообщается версия FreeBSD. Далее идентифицируется центральный процессор, включая его идентификационный номер, модель и тактовую частоту. Система сообщает также объем имеющейся памяти. Сюда входит как *реальная память* (заключенная в модулях SIMM, DIMM и т.д.), так и *доступная память* (реальная минус объем, занимаемый ядром, кэш-буферами BIOS и т.д.).

Большую часть сообщений ядра составляют сведения об аппаратных устройствах. Они, как правило, начинаются с названия устройства, за которым после двоеточия следует подробное описание. Некоторые отчеты занимают несколько строк, например:

```
npx0: <math processor> on motherboard
npx0: INT 16 interface
```

Здесь говорится о том, что на компьютере установлен математический сопроцессор. Это компонент, выполняющий операции с плавающей запятой. Все Intel-совместимые процессоры, начиная с 486DX, имеют встроенный сопроцессор. Более древние устройства, такие как 386, 486SX и некоторые процессоры NexGen, требуют чтобы микропроцессоры устанавливались в виде дополнительных микросхем на материнской плате.

Перечислим остальные устройства, часто упоминаемые в сообщениях ядра:

- **Шины PCI и ISA**. Устройства `pcib0`, `pci0`, `isab0` и `isa0` относятся к шинам PCI и ISA. *Шина* — это разъем, в который вставляется плата расширения.
- **Контроллеры ATA**. ATA (Advanced Technology Attachment) — это другое название интерфейса жестких дисков EIDE (Enhanced Integrated Drive Electronics). В современных компьютерах обычно присутствуют три ATA-устройства: контроллер ATA, установленный на материнской плате (`atapci0`) и две ATA шины (`ata0` и `ata1`).

- **Адаптеры SCSI.** Если в системе есть SCSI-адаптер, то для него будет создано логическое устройство. Его название зависит от набора микросхем, установленных на плате адаптера.
- **Контроллеры и устройства USB.** Устройства, названия которых начинаются на `u`, относятся к интерфейсу USB. Контроллеру USB соответствуют устройства `uhci0` и `ohci0`, устройство `usb0` реализует абстрактный интерфейс доступа к контроллеру, а устройство `uhub0` предоставляет доступ к USB-портам компьютера. Могут присутствовать и дополнительные устройства, например `umass0` (устройство массового хранения, такое как Zip-диск) или `ums0` (мышь).
- **Сетевые устройства.** Названия сетевых устройств, если таковые имеются, зависят от наборов микросхем. Обычно создается базовое логическое устройство, в описании которого указывается набор микросхем, и несколько дополнительных устройств, относящихся к конкретным компонентам сетевого интерфейса.
- **Клавиатура и мышь.** Если используется стандартная клавиатура, то для нее будут созданы логические устройства `atkbdc0` и `atkbd0`. Мыши соответствует устройство `psm0` (интерфейс PS/2), `ums0` (интерфейс USB) или другое устройство, в зависимости от типа интерфейса.
- **Устройства последовательного и параллельного доступа.** С параллельным портом (`ppc0`) может быть связано несколько подчиненных устройств: принтер (`lpt0`), общий параллельный интерфейс ввода-вывода (`ppi0`) или сетевой интерфейс (`plip0`). Последовательные порты (обычно `sio0` и `sio1`) могут поддерживать интерфейс PPP (`ppp0`), но обычно соответствующее устройство появляется только после того, как установлено PPP-соединение.
- **Консоль.** FreeBSD создает специальное логическое устройство `sc0` для доступа к консоли — обычно под этим понимаются клавиатура и монитор, подключенные непосредственно к компьютеру.
- **Видеоадаптер.** Для графической платы создается устройство `vga0`.
- **Дисковые устройства.** Отдельные устройства создаются для накопителей на гибких дисках (`fd0` и `fdc0`), жестких дисков EIDE (`ad0` и т.д.), жестких дисков SCSI (`da0` и т.д.) и дисководов CD-ROM (`acd0` для устройств EIDE). Иногда могут появляться сообщения о том, что не удалось определить емкость устройства для некоторых съемных накопителей, например Zip-дисководов. Это означает, что в момент загрузки системы в дисководе не было дискеты. На такие сообщения можно не обращать внимания.

Сообщения об ошибках, касающиеся вышеперечисленных устройств, могут указывать на то, что FreeBSD не удалось правильно распознать устройство или оно не работает. Если отсутствует важное устройство, значит, для него либо нет драйвера, либо оно повреждено. Некоторые устройства появляются не сразу. Это касается внешних устройств, которые на момент загрузки системы отключены. Иногда устройство помечается как `unknown`. Это означает, что оно найдено, но системе не удалось корректно его сконфигурировать. Так иногда бывает с платами расширения, для которых отсутствуют драйверы.

Сценарии запуска системы

После загрузки ядра вызывается программа `init`. Она управляет запуском системы посредством сценариев, находящихся в каталоге `/etc`. Их имена начинаются с префикса "`rc`". Эти сценарии запускают серверы и другие программы, выполняющиеся в фоновом режиме. Многие из сценариев генерируют собственные сообщения, кото

рые, однако, не сохраняются и не могут быть просмотрены с помощью команды dmesg, в отличие от сообщений ядра

Перечислим основные задачи программы init и сценариев запуска системы

- **Проверка файловых систем**. При запуске FreeBSD определяет, *чисты* ли дисковые разделы, т.е. корректно ли они демонтированы. Если это не так, запускается программа fsck, проверяющая разделы и исправляющая ошибки, которые могли быть результатом сбоя или некорректного останова системы.
- **Монтирование разделов**. FreeBSD монтирует разделы, руководствуясь установками файла /etc/fstab (описан в главе 7). В нем хранится информация о разделах и точках монтирования, заданная при инсталляции FreeBSD.
- **Конфигурирование**. FreeBSD должна выполнить множество различных задач, связанных с инициализацией системы, например сконфигурировать сетевые интерфейсы
- **Запуск серверов и других программ**. Некоторые программы должны выполняться непрерывно все то время, пока работает система. Сюда входит большинство серверов, и не только они. Такие программы запускаются автоматически на этапе загрузки системы

Большинство серверов функционирует в качестве *демонов*. Слово "daemon" заимствовано из греческого языка и означает "помощник". Во FreeBSD (и вообще в UNIX) демоны работают в фоновом режиме без вмешательства пользователей, реализуя необходимые функции. Многие демоны являются серверами, хотя и не все. Омонимия слов "daemon" и "demon" является одной из причин того, что символом FreeBSD был выбран красный чертенок

- **Разрешение регистрации в системе**. Чтобы можно было войти в систему, на компьютере должна выполняться регистрационная программа, принимающая от пользователя имя и пароль и выполняющая его авторизацию. Одни регистрационные программы являются сетевыми серверами, другие работают локально. Запуск последних контролируется установками файла /etc/ttys.

Сценарии запуска системы будут более подробно описаны в главе 6. Пока же достаточно знать, что с их помощью вызывается множество программ, в том числе для регистрации в системе

Регистрация в системе

По завершении загрузки системы появляется регистрационное приглашение. Чтобы войти в систему, нужно ввести имя пользователя и пароль. Но прежде давайте выясним, чем отличается регистрация в текстовом и графическом режимах и в чем разница между обычными учетными записями и учетной записью root. Нужно также понимать, что происходит после регистрации и как стать суперпользователем, войдя в систему от имени рядового пользователя

Регистрация в текстовом и графическом режимах

По умолчанию во FreeBSD обеспечивается регистрация в текстовом режиме. Это означает, что после загрузки системы на экране появится приглашение следующего вида:

```
login:
```

Таким образом система спрашивает имя пользователя Введите его и нажмите <Enter> Сразу же появится следующее приглашение

```
Password.
```

Введите пароль и снова нажмите <Enter> Из соображений безопасности пароль не отображается на экране. Если введен правильный пароль, FreeBSD отобразит привет ственную информацию а в нижней части экрана появится *приглашение командной строки*. Обычно это знак доллара ($) Режим суперпользователя помечается знаком решетки (#). Впрочем, эти установки можно менять.

В текстовом режиме можно запускать соответствующие программы и вводить тек стовые команды как будет описано ниже Иногда возникает необходимость запустить сразу несколько программ. Это можно сделать двумя способами.

- **Виртуальные консоли.** FreeBSD поддерживает механизм *виртуальных консолеи* позволяющий пользователю регистрироваться несколько раз. Для этого нужно нажать <F2> или любую другую функциональную клавишу вплоть до <F8> Появится еще одно приглашение `login:`, и можно будет зарегистрироваться второй раз Чтобы вернуться к первой виртуальной консоли, нажмите <F1> На разных экранах можно запускать разные программы. Разрешается даже регистрироваться от имени другого пользователя Если запущен X-сервер то для переключения между консолями нужно нажимать клавишу <Alt> вместе с соответствующей функциональной клавишей Самому серверу соответствует девятая консоль
- **Фоновая обработка.** Программу можно запустить в фоновом режиме, когда она перестает принимать данные с клавиатуры. Для этого нужно к строке вызова программы добавить оператор &, например `numbercrunch &` В качестве альтернативы можно просто приостановить выполнение программы, нажав <Ctrl+Z> Программа не будет уничтожена зато снова появится командная строка Это особенно удобно, если нужно временно выйти из программы, а за тем вернуться в нее Для возврата в программу следует ввести `fg` Команда `bg` запускает программу в фоновом режиме

В текстовом режиме нельзя вызывать графические программы, не запустив пред варительно X Window System, как будет описано ниже Можно сконфигурировать FreeBSD так чтобы X-сервер запускался автоматически на этапе загрузки системы В такой системе есть программа наподобие XDM (X Display Manager), отображающая запрос на регистрацию (рис 3 1) Если ввести имя пользователя и пароль, FreeBSD перейдет в графический режим Программа XDM должна вызываться из одного из сценариев запуска системы Необходимо также настроить параметры регистрации от дельных пользователей В главе 21 описано, как конфигурируются XDM и аналогич ные программы а в главе 6 рассматриваются файлы, управляющие регистрацией пользователей

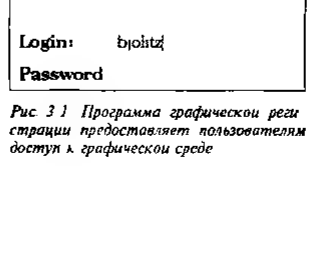


Рис 3 1 Программа графическои реги страции предоставляет пользователям доступ к графическои среде

Обычные и суперпользовательские учетные записи

Как описывалось в главе 2, во FreeBSD изначально есть две учетные записи: для рядового пользователя и root. Первая из них предназначена для повседневной работы: чтения электронной почты, просмотра Web-страниц, обработки графических файлов и т.д. Учетная запись root применяется системным администратором, который называется *суперпользователем*. Она предоставляет практически неограниченный доступ к компьютеру. Пользователь root может читать, записывать, перемещать и удалять любые файлы. Для решения большинства задач права суперпользователя не нужны. Они требуются лишь при выполнении административных действий.

На заметку В этой книге вводимым командам предшествует приглашение командной строки. Для рядового пользователя это будет знак доллара ($), а для пользователя root — знак решетки (#). Таковы установки по умолчанию, принятые в системе.

Иногда после регистрации в системе возникает необходимость выполнить разовую привилегированную операцию. Эту проблему можно решить несколькими способами. Один из них — воспользоваться другой виртуальной консолью для регистрации под именем root, выполнить необходимое действие, а затем выйти из этого режима. Но такой подход неудобен, если нужно в графической среде наряду с обычными программами запустить программу с привилегиями суперпользователя. Впрочем, даже в текстовом режиме существует более простое средство — команда su.

Название команды расшифровывается как "substitute user", т.е. "поменять пользователя". Она позволяет изменить идентификатор пользователя, уже зарегистрировавшегося в системе. Синтаксис вызова команды таков:

```
su [ [ flm] [-c класс] [имя пользователя [аргументы]]
```

На заметку Такой способ синтаксической записи часто применяется в документации к FreeBSD. Квадратные скобки обозначают необязательный параметр. Односимвольные опции разрешается объединять, например -flm, или записывать раздельно, например -f, -l. Курсивом выделяются переменные аргументы. Троеточие (здесь не показано) говорит о том, что опция может повторяться. Вертикальная черта (, тоже не показана) разделяет два альтернативных варианта, которые не могут использоваться одновременно.

Команду su можно вызывать без аргументов. Если пользователь, который вводит такую команду, является членом группы wheel, команда попросит ввести пароль суперпользователя. Введя правильный пароль, пользователь получит расширенные привилегии. Если же указать имя конкретного пользователя, команда попросит ввести его пароль. По окончании работы под другим именем введите exit, чтобы вернуться в обычный режим. Ниже описаны параметры команды su.

- **-l или одиночный дефис (-)**. Обычно команда su создает среду, которая не совсем идентична традиционной регистрационной среде. Опция -l (или просто -) отменяет большинство текущих установок пользовательской среды и формирует соответствующую среду целевого пользователя.
- **-f**. Если целевой пользователь работает с интерпретатором команд csh, то данная опция запрещает команде su читать конфигурационный файл интерпретатора (.cshrc). Тем самым можно сделать целевую среду меньше обычного.
- **-m**. Эта опция позволяет поменять идентификатор пользователя без изменения среды. К примеру, можно применять привычный интерпретатор команд, даже если целевой пользователь работает с другим.

- *-c класс*. FreeBSD поддерживает *классы* регистрации напоминающие группы пользователей. Можно задать класс, установки которого требуется использовать
- *аргументы*. Если требуется передать аргументы интерпретатору команд целевого пользователя, их нужно указать в конце строки вызова команды su

Команда su очень удобна для решения административных задач. Можно зарегистрироваться в системе как рядовой пользователь, а затем по мере необходимости запускать привилегированные программы с помощью команды su. Она доступна как в текстовом, так и в графическом режиме (при условии, что открыто окно для ввода команд)

С консоли можно зарегистрироваться непосредственно под именем root, но в качестве меры предосторожности FreeBSD не принимает такие запросы по сети через программу Telnet Это ограничение означает, что злоумышленник, узнавший пароль суперпользователя, должен будет выяснить еще чей-нибудь пароль для вторжения в систему Ему придется сначала войти в систему от имени рядового пользователя, а затем выполнить команду su, чтобы стать суперпользователем. То же самое справедливо и в отношении администраторов, что вызывает определенные неудобства. Но эти неудобства компенсируются повышением безопасности системы Более того, даже при работе за консолью нежелательно регистрироваться под именем root. Если пользоваться командой su, FreeBSD будет фиксировать факт ее вызова, помечая имя пользователя, который выполнил команду su Такая информация позволяет контролировать применение административных полномочий и оперативно выявлять нарушителей

Команды текстового режима

Поскольку FreeBSD по умолчанию запускается в текстовом режиме, мы сначала опишем утилиты текстового взаимодействия В первую очередь будут рассмотрены *интерпретаторы команд* — программы, принимающие команды от пользователей Важно понимать правила работы интерпретаторов, в частности то, как они воспринимают имена файлов и какие программы могут запускать

Те, кто привык работать в графической среде, такой как Windows или Mac OS, могут посчитать, что этот материал можно пропустить Но многие административные действия во FreeBSD удобнее выполнять в текстовом, а не графическом режиме Текстовый режим часто отпугивает отсутствием привычных средств интерфейсного взаимодействия, однако его возможности очень велики, особенно когда речь идет о FreeBSD, а не Windows Таким образом, изучение средств текстового взаимодействия необходимо для понимания возможностей системы.

Интерпретаторы команд

Когда пользователь регистрируется в системе в текстовом режиме, FreeBSD запускает программу, называемую интерпретатором команд Как описывалось в главе 2, название интерпретатора задается в параметрах учетной записи В принципе, интерпретатором может служить любая программа, но только программы работающие по определенным правилам, задействуются при регистрации в системе Стандартным интерпретатором является sh (Bourne shell). Следует упомянуть также csh (во FreeBSD реализован как tcsh), bash, ksh и zsh. Все они являются обычными программами FreeBSD. Одни находятся в каталоге /bin, другие — в каталоге /usr/local/bin Большей частью все они работают одинаково хотя и имеют разные

функциональные возможности а также некоторые мелкие отличия Например, по умолчанию интерпретаторы выдают разное приглашение командной строки, а также используют разные конфигурационные файлы. У sh это файл .shrc, находящийся в начальном каталоге пользователя, у tcsh — .tcshrc или .cshrc, у bash — .bashrc Форматы этих файлов тоже отличаются, поэтому нельзя просто переименовать, к примеру, .shrc в .tcshrc и рассчитывать, что все заработает Некоторые из этих файлов кратко описаны в главе 6.

Помимо традиционных интерпретаторов команд UNIX, FreeBSD поддерживает также ряд нестандартных интерпретаторов. Один из них — mudsh, напоминающий старые текстовые игры со встроенными командами и приглашениями. Другой — pash, реализующий текстовый интерфейс диспетчера файлов наподобие Norton Commander для DOS (рис. 3.2)

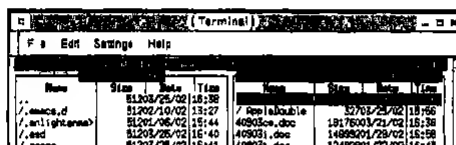


Рис 3.2. Интерпретаторы команд могут работать как в текстовом режиме, так и в графическом окне

Если хотите поэкспериментировать с различными интерпретаторами, инсталлируйте их с помощью программы sysinstall (раздел configure) Для запуска интерпретатора достаточно ввести его имя Чтобы сменить стандартный интерпретатор команд, необходимо модифицировать параметры учетной записи, как будет описано в главе 10

Все интерпретаторы поддерживают встроенные и внешние команды. Большинство команд, вызываемых непосредственно из интерпретатора, являются внешними. Внутренние команды удобно использовать в сценариях, которые представляют собой не большие программы написанные на языке интерпретатора команд. В сценариях можно запускать другие программы, манипулировать файлами и т.д. (см главу 31).

Некоторые интерпретаторы, включая tcsh и bash, поддерживают возможность *дополнения команд*: можно ввести неполную команду и нажать <Tab>, а интерпретатор корректно завершит командную строку. Реакция интерпретатора зависит от того, что именно введено

- Если ввести часть имени программы интерпретатор попытается найти эту программу Например, если ввести chm и нажать <Tab>, интерпретатор, скорее всего, предложит вариант chmod
- Если ввести полное имя программы и часть имени файла, передаваемого ей в качестве аргумента, интерпретатор попытается найти файл по заданному шаблону

Когда интерпретатор обнаруживает несколько совпадений, он подает звуковой сигнал или отображает все возможные варианты. В случае звукового сигнала нажмите <Tab> еще раз. Если снова прозвучит звуковой сигнал, значит, совпадений не найдено.

В режиме дополнения не распознаются аргументы команд, за исключением имен файлов. Зато можно одновременно задать несколько файлов. Удобными подстановочными знаками являются звездочка (*) и знак вопроса (?). Они воспринимаются примерно так же, как в DOS или Windows, и могут присутствовать в именах файлов, но не программ: последние должны быть указаны точно Звездочка обозначает любой символ или набор символов, включая полное их отсутствие Например, выражению `F*D` соответствуют имена `FD`, `FeD`, `FreeBSD` и любые другие, начинающиеся на `F` и заканчивающиеся на `D`. Знак вопроса обозначает ровно один символ поэтому выражению `F?D` соответствуют имена `FeD` и `FFD`, но не `FD` или `FreeBSD`.

Во FreeBSD есть и другие подстановочные знаки. Например, группы искомых символов часто помещаются в квадратные скобки. Если в данной позиции слова встречается один из символов указанный в скобках, то оно соответствует шаблону поиска Таким образом, выражению `F[aeu]D` соответствуют имена `FaD`, `FeD` и `FuD`, но не `FooD` или `FeeD`. Вместо отдельных символов разрешается указывать символьный интервал например `F[a-c]D` Этому выражению соответствуют имена `FaD`, `FbD` и `FcD`.

Запуск программ в текстовом режиме

Для запуска программы достаточно указать ее имя в командной строке Например, чтоб запустить программу `someprogram`, введите **`someprogram`** и нажмите <Enter> Если программа находится там, где интерпретатор команд обычно ищет программы, она будет загружена в память и запущена В зависимости от назначения, программа может управлять экраном, отображать данные, требовать ввода данных и т д Многие UNIX-программы ничего не выводят на экран в случае корректной работы Только при возникновении ошибок на экране появляются сообщения.

После ввода имени программы интерпретатор начинает искать ее в нескольких каталогах Они образуют *путь поиска*, который задается с помощью переменной среды `PATH` в файлах конфигурации интерпретатора (как персональных, так и общесистемных) Обычно в путь поиска входят каталоги `/bin`, `/usr/bin`, `/usr/X11R6/bin`, `/usr/local/bin` и некоторые другие Для пользователя `root` путь поиска включает каталоги `/sbin` и `/usr/sbin`. В них хранятся программы, предназначенные для запуска системным администратором, а не рядовым пользователем. У суперпользователя путь поиска короче обычного, что затрудняет вызов многих программ. Это лишний раз доказывает, что не все действия нужно выполнять в режиме суперпользователя.

Если программа не найдена, нужно указать полный путь к ней, например `/opt/bin/someprogram` Существует также понятие *относительного пути* Это значит, что местонахождение файла задается относительно текущего каталога. Относительный путь не начинается с обратной косой черты, а задается одним из следующих способов.

- Имя может начинаться с удвоенной точки (`..`), что указывает на родительский каталог. К примеру, каталог `/home` будет родительским для `/home/bjolitz`. Если, находясь в каталоге `/home/bjolitz`, ввести `../../opt/bin/someprogram`, то это будет эквивалентно записи `/opt/bin/someprogram`.
- Можно задать подкаталог текущего каталога. Например, в каталоге `/opt` достаточно ввести `bin/someprogram` чтобы запустить программу `/opt/bin someprogram`
- Префикс `./` означает, что программа находится в текущем каталоге. Иногда запись '`.`' присутствует в пути поиска, тогда этот префикс можно опустить

Внимание

Рискованно включать спецификацию в путь поиска, поскольку яю бая программа в текущем каталоге будет выполняться так, как если бы она находилась в одном из каталогов стандартных программ, доступ ных для записи только суперпользователю. В крайнем случае включите спецификацию в конец пути поиска, чтобы интерпретатор просматри вал его в последнюю очередь. Это особенно важно для пользователя `root`, так как злоумышленники часто пытаются обмануть его, заставив выполнять неавторизованные программы из своих начальных катало гов. Лучше всего вообще обойтись без включения начального каталога в путь поиска

- Начальная тильда (~) указывает на начальный каталог текущего пользователя

Эти правила применимы к любым спецификациям файлов, а не только к вызовам программ. При ссылке на файл данных, находящийся в текущем каталоге, префикс / опускается. Поэкспериментируйте со спецификациями при помощи команд `cd` (переход в новый каталог) и `ls` (вывод содержимого текущего или указанного ката лога). Если вы "заблудились" в системе, введите `cd` чтобы вернуться в начальный ка талог, или `pwd`, чтобы узнать, где вы находитесь.

Многие программы принимают вспомогательные *опции*, *параметры*, *аргументы* т е. информацию, сообщаемую программе в командной строке. Например, можно указать редактору имя загружаемого файла. В основном дополнительные параметры модифицируют работу программы.

Новички должны знать, что FreeBSD, как и другие разновидности UNIX, *чувствительна к регистру символов*, т е. различает строчные и прописные буквы в именах программ и файлов. Скажем, если ввести `pwd`, система сообщит имя текущего каталога, но если ввести `PWD` то появится сообщение об ошибке, например `command not found`. Опции большинства команд тоже различаются в зависимости от регистра, хотя бывают и исключения.

Команды манипулироввния файлами

С помощью описываемых ниже команд можно выполнять базовые операции нал файлами, например выводить список файлов в каталоге, перемещать файлы и удалять их. Существует множество других утилит, описываемых в различных местах книги. Так, в главе 5 рассматриваются текстовые редакторы, а в главе 8 — команды позволяющие изменить принадлежность файлов и права доступа к ним.

cd изменение текущего каталога

Команда `cd` переводит пользователя в другой каталог. Например, если в начальном каталоге есть подкаталог `taxes`, то для перехода в него нужно ввести `cd taxes` (или `cd /taxes`, если вы находитесь не в начальном каталоге). Необходимость менять те кущий каталог возникает не так уж часто, просто в некоторых случаях это позволяет упростить ввод последующих команд и не указывать лишние каталоги.

Команда `cd` является встроенной функцией интерпретатора и, в отличие от боль шинства команд, не является отдельной программой.

ls списки файлов

Если нужно узнать, какие файлы находятся в текущем каталоге, воспользуйтесь командой `ls`. Результаты ее работы будут примерно такими:

```
$ ls
GNUstep          Ma l              XF86Config new  xin t core
```

По умолчанию для каждого файла указывается только имя. Но команда ls спо собна собирать гораздо больше информации о файлах. При наличии опции -l коман да выдает *расширенный* листинг:

```
$ ls -l
total 406
drwxr-xr-x  5 rodsmith  users     512 Dec 15 22:25 GNUstep
drwx------  2 rodsmith  users     512 Jan  1 20:51 Mail
-rw-r--r--  1 rodsmith  users    3263 Dec 15 21:35 XF86Config.new
-rw-------  1 rodsmith  users 397312 Dec 15 20:50 xinit.core
```

Здесь указываются права доступа к файлу (называемые *режимом* и отображаемые в виде набора из десяти символов в начале каждой строки), имя владельца (rodsmith), группа, которой принадлежит файл (users), его размер, а также дата и время созда ния (если файл был создан больше года назад, будет указан год, а не время создания).

Если команде ls задать имя каталога, она отобразит информацию о файлах в нем. А если воспользоваться подстановочными знаками, то команда сообщит сведения обо всех файлах или каталогах, соответствующих шаблону. Например, команда ls *.txt выдаст список файлов с расширением .txt.

Команда ls поддерживает большое число различных опций. Подробная информа ция о них содержится на man-странице команды (введите **man ls**, чтобы просмотреть ее). Ниже описано несколько полезных опций.

- **-a**. Заставляет команду отобразить список всех файлов каталога. По умолча нию не отображаются так называемые *файлы с точкой*, имена которых начи наются с точки, например .tcshrc. Как правило, это конфигурационные фай лы программ с аналогичными именами без точки (tcsh в случае файла tcshrc).
- **-R**. Включает режим рекурсивного листинга, в котором выдается информация не только о заданном каталоге, но и обо всех его подкаталогах. Учтите, что список может получиться огромным, поэтому иногда приходится направлять результаты через канал программе постраничной разбивки, например ls -R /usr | less.
- **F**. По умолчанию в режиме короткого листинга команда ничего не сообщает о типе каждого файла. При наличии же этой опции команда будет отображать после некоторых файлов специальный символ, обозначающий тип файла: / — каталог, * — программа, @ — символическая ссылка. Есть и другие обозначения.

На заметку *Символическая ссылка* — это указатель на файл или каталог, хранящийся под другим именем и/или в другом каталоге. Такие ссылки занимают немного места на диске, зато позволяют обращаться к файлам и ка талогам под несколькими именами.

- **-f**. По умолчанию команда ls сортирует списки в алфавитном порядке. Эта опция отключает сортировку.
- **-n**. Используется в сочетании с опцией -l. Благодаря ей вместо имен пользо вателя и группы подставляются соответствующие идентификаторы.

Можно объединять несколько опций в один строковый аргумент, а также указы вать каталог, содержимое которого требуется просмотреть. К примеру, следующая ко манда выводит короткий список всех файлов в каталоге /tmp с указанием их типа:

```
$ ls -aF /tmp
      X0-lock      sawfish-rodsmith/  sample-file.txt@  temprog
./    X11-unix/    orbit-rodsmith/    sample.txt
```

cp. копирование файлов

Команда cp копирует файлы Синтаксис ее вызова таков

```
cp [опции] исходный_файл целевой_файл
cp [опции] исходный_файл целевой_каталог
```

В первом случае указывается точное имя файла после копирования Оно может включать в себя путь к файлу, если он копируется в другой каталог. Во втором случае файл копируется в другой каталог под тем же именем

Как и ls, команда cp поддерживает большое число опций, позволяющих выполнять рекурсивное копирование (т е копировать все дерево каталогов), обрабатывать символические ссылки и т д Обо всем этом рассказывается в главе 8

mv' перемещение файлов

Команда mv перемещает файл в другои каталог и/или меняет его имя Ее синтаксис такой же, как и у команды cp.

Те, кто знакомы с DOS или Windows, должны учитывать, что команда mv выполняет функции двух отдельных DOS-команд MOVE и RENAME. Если указать целевой каталог, команда mv будет работать подобно DOS-команде MOVE перемещая файл в другой каталог. Если же указать имя целевого файла, то команда mv поменяет его имя. Оба действия могут быть совмещены в одной команде. Рассмотрим следующие примеры

```
$ mv file.txt /tmp
$ mv file.txt /tmp/somefile.txt
$ mv file.txt somefile.txt
```

Первая команда копирует файл file.txt в каталог /tmp без изменения имени Вторая копирует и переименовывает файл, а третья - переименовывает файл в текущем каталоге. Если указывается только целевой каталог, то разрешается перемещать несколько файлов, перечисляя их поименно или включая в состав спецификации метасимволы, например

```
$ mv file.txt morefile? txt /tmp
```

Эта команда копирует в каталог /tmp файл file txt а также любой фаил соответствующий шаблону morefile?.txt

При перемещении файла из одного каталога в другой в пределал одного раздела диска или съемного устройства, не происходит повторная запись файла. Вместо этого в новом каталоге создается ссылка на файл, а старая ссылка удаляется. Таким образом, перемещение осуществляется очень быстро. Если же выполняется смена раздела FreeBSD вынуждена копировать исходный файл, а затем удалять его Это происходит чуть медленнее, хотя разницу можно заметить лишь для очень больших файлов

rm удаление файлов

Если файл больше не нужен, его можно удалить с помощью команды rm. Метасимволы позволяют удалять несколько файлов одноп командой Подробнее опции команды rm описываются в главе 8. Они дают возможиость удалить все дерево каталогов, запросить подтверждение перед удалением и даже очистить содержимое файлов перед удалением

Внимание

По умолчанию команда rm не запрашивает подтвержденне перед удалением файлов Это делает ее очень опасным средством, особеино в руках суперпользователя Указав опцию рекурсивного удаления, можно одной командой удалить все файлы FreeBSD' Конечно, команда rm необходима но всегда заранее проверяйте что вы собираетесь удалять

mkdir: создание каталога

FreeBSD работает с каталогами примерно как с файлами. По сути, каталоги и есть файлы, в которых содержатся имена других файлов и ссылки на них. Для создания каталогов предназначена отдельная команда `mkdir`, синтаксис которой таков:

```
mkdir [-pv] [-m режим] имя_каталога
```

Например, команда `mkdir textfiles` создает подкаталог `textfiles` в текущем каталоге. Имя каталога может включать путь к нему. Ниже описаны опции команды:

- **`-p`** По умолчанию команда `mkdir` требует, чтобы существовал непосредственный родительский каталог заданного каталога. Например, если ввести `mkdir /dir1/dir2`, а каталог `~/dir1` не существует, команда `mkdir` выдаст сообщение об ошибке. Опция `-p` заставляет команду создать все необходимые каталоги.
- **`v`** Эта опция включает режим расширенных сообщений, в котором команда сообщает о каждом создаваемом каталоге.
- **`m`** *режим* С каждым файлом и каталогом связан набор прав доступа, называемый *режимом*. Эта опция позволяет задать режим доступа к каталогу (глава 8).

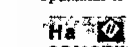
rmdir: удаление каталога

Если нужно удалить каталог, воспользуйтесь командой `rmdir`, указав имя требуемого каталога. Он должен быть пустым, иначе команда откажется удалять его. Если перед именем каталога поставить опцию `-p`, команда удалит не только его, но и все родительские каталоги, указанные в качестве пути к нему. Например, команда `rmdir -p dir1/dir2` удаляет каталог `dir2`, а затем — `dir1` при условии, что оба они пусты.

Чтобы удалить все дерево каталогов, включая непустые каталоги, воспользуйтесь опцией `-r` команды `rm`, как описано в главе 8.

Краткий обзор графической среды

Возможности текстового режима FreeBSD довольно велики. В некоторых специализированных серверах даже не приходится устанавливать графическую среду — X Windows System (сокращенно X-среда). Но в большинстве рабочих станций она нужна, и нужно привыкнуть работать в ней. В этом разделе рассказывается о том, как запустить графическую оболочку, как манипулировать файлами в ней, запускать программы и менять ее установки.



X Window System — невероятно гибкая графическая среда. В зависимости от параметров инсталляции и персональных установок, может оказаться, что на другом компьютере среда выглядит совсем не так, как описано здесь. Подробнее параметры конфигурации среды описаны в главе 13.

Переход в графический режим

Если по окончании загрузки FreeBSD появляется графическое приглашение на регистрацию (см. рис. 3.1), то для запуска X-сервера ничего не нужно, он уже работает, необходимо лишь зарегистрироваться в системе. Но этот режим не включен во

FreeBSD по умолчанию так как предпочтение отдается текстовому режиму Чтобы изменить эту установку, нужно запустить XDM или другую аналогичную программу, как рассказывается в главе 21 Но в графический режим можно перейти и непосредственно из текстового с помощью команды `startx` Будет запущен X-сервер и загружена стандартная графическая среда, параметры которой заданы на этапе инсталляции системы (стандартные установки хранятся в файле `.xinitrc`).

На заметку Некоторые пользователи добавляют к команде `startx` опции, включающие режим с необходимыми разрешением и цветовой разрядностью Но достаточно один раз настроить конфигурационные файлы чтобы этого не пришлось делать

Если все пройдет успешно, вместо текстового экрана появится графический на чальное содержимое которого зависит от выбранных установок Он может быть пустым или содержать несколько окон На рис 3.3 представлен экран графической оболочки GNOME Именно на нее ориентировано последующее описание Различные графические оболочки во многом похожи хотя многие детали, конечно, будут отличаться

На заметку Если X-сервер не запустился корректно, просмотрите последние несколько сообщений и обратитесь к главам 13 и 32

Начнем с изучения возможностей диспетчера окон и самой графической оболочки Большинство диспетчеров окон FreeBSD формируют интерфейс, напоминающий Windows или Mac OS Подчеркнем ряд важных различий между многочисленными диспетчерами и оболочками

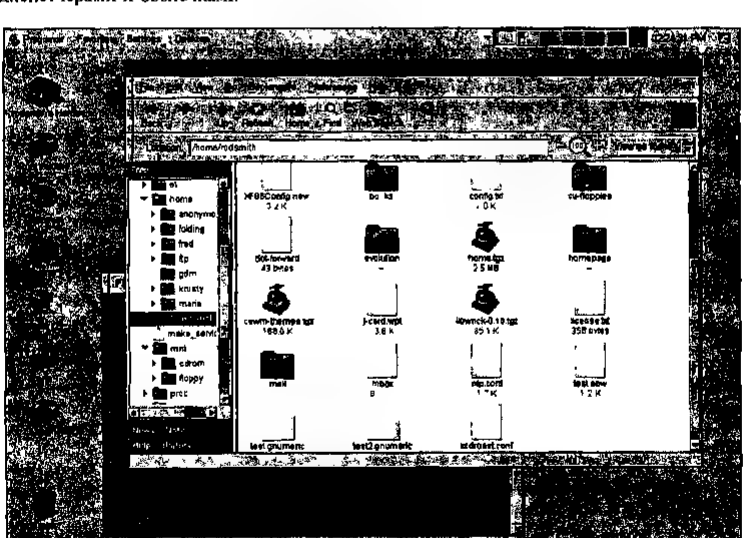


Рис 3 3 Графическая оболочка GNOME напоминает оболочки Windows и Mac OS

- **Фокус окна** Говорят что окно *имеет фокус*, когда оно может принимать данные с клавиатуры или реагировать на щелчки мыши. У такого окна строка заголовка выделена другим цветом Обычно окно, имеющее фокус, находится на *переднем плане*, т.е. не перекрыто никаким другим окном Например, на рис. 3.3 окно диспетчера файлов (заголовок rodsmith) находится на переднем плане и имеет фокус Большинство диспетчеров окон перемещает окно на передний план и передает ему фокус при щелчке мышью в любой части окна Иногда для этого достаточно навести указатель мыши на окно
- **Элементы управления** *Элемент управления* – это графическая заготовка, позволяющая определенным образом взаимодействовать с программой. Как правило, элементы управления диспетчера окон расположены в левой и правой части строки заголовка окна Работают они в основном так же, как и в Windows
- **Контекстные меню.** Большинство диспетчеров окон создает меню с элементами касающимися запуска программ, завершения работы диспетчера и т.д Для вызова такого меню нужно щелкнуть правой кнопкой мыши на рабочем столе В других графических оболочках, включая GNOME, такое меню расположено в углу экрана. В частности, в GNOME для вызова меню нужно щелкнуть на значке с изображением ступни в нижнем левом углу экрана
- **Меню и панели задач.** В большинстве графических оболочек создаются меню зафиксированные вдоль верхней, нижней или боковой части экрана. На рис. 3 3 такие меню (называемые *панелями задач*) изображены вверху и внизу Верхняя панель позволяет задавать параметры оболочки GNOME и переключаться между виртуальными рабочими столами (это виртуальные экраны, на которых можно запускать программы, как в случае виртуальных терминалов текстового режима). Нижняя панель содержит стартовое меню (значок с изображением ступни), предназначенное для запуска программ, а также значки быстрого вызова важных программ и кнопки переключения между окнами программ, работающих в данный момент в пределах рабочего стола

Если вследствие экспериментов была повреждена рабочая среда, завершите работу X-сервера и удалите пользовательские конфигурационные файлы из начального каталога В случае оболочки GNOME нужно удалить каталоги, имена которых начинаются на `.gnome`, включая `gnome` и `gnome-desktop`. При следующем запуске X-сервера оболочка GNOME восстановит стандартные установки

Управление файлами

Специальная программа, называемая диспетчером файлов (окно rodsmith на рис. 3.3), позволяет перемещать и копировать файлы, а также выполнять другие манипуляции над ними Как правило, на рабочем столе присутствует значок *пользователь* Home Выполните двойной щелчок на нем, чтобы вызвать диспетчер файлов. Если этот значок отсутствует, то в оболочке GNOME нужно в стартовом меню выбрать команду Programs⇨Applications⇨Nautilus или же в терминальном окне (на рис 3 3 перекрывается окном диспетчера) ввести `nautilus`.

Чтобы перейти к нужному каталогу, щелкните на вкладке Tree в левой части окна диспетчера и в появившемся списке выберите каталог. Если требуется выбрать подкаталог, раскройте список каталогов, щелкнув на значке треугольника слева от названия каталога. Иногда удобно иметь на экране два окна. Чтобы открыть новое окно, выберите команду File⇨New Window или снова щелкните на значке *пользователь* Home Ниже перечислены основные возможности диспетчера файлов.

- **Копирование файлов** Щелкните правой кнопкой мыши на значке файла и выберите в появившемся меню команду Duplicate. Копия будет иметь то же имя но с добавлением префикса (copy). Можно также щелкнуть на значке файла и перетащить его в нужный каталог в том же самом или в другом окне. В результате появится меню, позволяющее переместить или скопировать файл, создать символическую ссылку на него или отменить операцию.
- **Перемещение файлов**. Чтобы переместить файл, щелкните на его значке правой кнопкой мыши и перетащите его, как при копировании.
- **Удаление файлов** Щелкните правой кнопкой мыши на значке файла и выберите в появившемся меню команду Move to Trash, чтобы переместить файл в "мусорную корзину" (значок Trash). Можно также переместить значок файла непосредственно на значок Trash. В результате файл окажется в специальном "мусорном" каталоге, который впоследствии можно очистить, выбрав в меню диспетчера файлов команду File⇨Empty Trash. Если окажется, что файл нужно восстановить, это будет несложно сделать.
- **Переименование файлов**. Чтобы переименовать файл, щелкните на его значке правой кнопкой мыши и выберите в появившемся меню команду Rename. То же самое можно сделать и в окне свойств файла, которое вызывается с помощью команды Show Properties (рис. 3.4).
- **Изменение прав доступа**. Чтобы поменять права доступа к файлу, щелкните правой кнопкой мыши на его значке, выберите команду Show Properties, перейдите в окне свойств на вкладку Permissions и задайте нужные права.
- **Запуск программ**. Если программа GNOME закрепила за каким-либо типом файлов определенное приложение, то для вызова этого приложения достаточно выполнить двойной щелчок на значке файла соответствующего типа в окне диспетчера. Если нужно загрузить файл в другое приложение, щелкните на его значке правой кнопкой мыши и перейдите в меню Open With. В нем будут перечислены доступные приложения.

В оболочке KDE есть схожий диспетчер файлов, одновременно являющийся Web броузером. FreeBSD поддерживает множество других диспетчеров файлов, которые можно использовать независимо от конкретных графических оболочек.

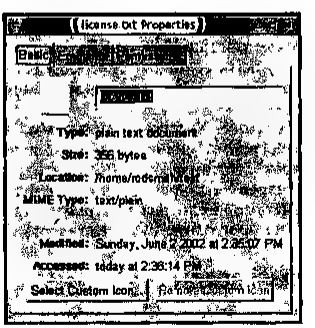


Рис. 3.4. В окне свойств можно переименовать файл, изменить группу, которой он принадлежит, и права доступа к нему

Запуск программ в графической среде

Существует несколько способов запуска программ в графической среде

- **Значки программ на панели задач** Значки, расположенные справа от значка стартового меню на рис. 3.3, предназначены для быстрого вызова четырех часто используемых программ GNOME Help Browser (отображает справку) GNOME Terminal (предоставляет доступ к командной строке) GNOME Control Center (описана ниже) и Mozilla (Web-броузер)
- **Стартовые меню** Стартовое меню в нижней части экрана и меню Programs в верхней части позволяют получать доступ к программам, установленным в системе
- **Диспетчер файлов.** Программу можно запустить и через диспетчер файлов, вы полнив двойной щелчок на значке программного файла или файла данных тип которого ассоциирован с программой.
- **Значки на рабочем столе** Некоторые значки, расположенные на рабочем столе предназначены для запуска программ Большинство таких значков на рис. 3 3 служит для открытия окон диспетчера Nautilus но можно добавить новые значки
- **Терминал.** Запустив программу Terminal, можно перейти в режим командной строки и ввести в терминальном окне имя требуемой программы В терминальном окне выполняется стандартный интерпретатор команд текстового режима (`sh`, `tcsh`, `bash` или другой)

На заметку

Программа Terminal относится к классу программ, используемых для запуска текстовых интерпретаторов и других текстовых программ в графической среде Наиболее универсальная из программ этого класса называется `xterm`, и иногда ее именем обобщенно называют все такие программы

Графические программы запускаются в отдельных окнах В то же время программа, вызываемая из терминального окна, продолжает "владеть" им вследствие чего терминал становится недоступным, если только не вызвать программу с аргументом `&`, который переведет ее в фоновый режим. Текстовые программы обычно работают непосредственно в окне терминала

Поскольку оконные оболочки существенно отличаются друг от друга в этой книге мы сосредоточимся на запуске программ из среды `xterm`. Конечно, если программа доступна через меню, то запускать ее таким способом гораздо проще Можно также создать на рабочем столе значки для быстрого вызова наиболее часто используемых программ.

Особенностью FreeBSD (точнее, X-среды и программ, написанных для нее) является то, что графические программы сильно отличаются по внешнему виду. X-среда состоит из нескольких "слоев", начиная от X-сервера и заканчивая оболочкой рабочего стола Одни из этих "слоев" выбираются пользователями (например, диспетчер окон), другие — программистами Если два программиста выбирают разные наборы инструментальных средств, их программы будут выглядеть по-разному

Настройка графической среды

Некоторые элементы графической среды приходится почти сразу же переконфигурировать Например, пользователи любят помещать на рабочий стол фоновые фотографии. В разных графических оболочках способы настройки тоже разные. В среде GNOME есть программа Control Center (рис. 3.5), вызываемая из панели задач в нижней части экрана. Можно также ввести `gnomecc` в окне терминала

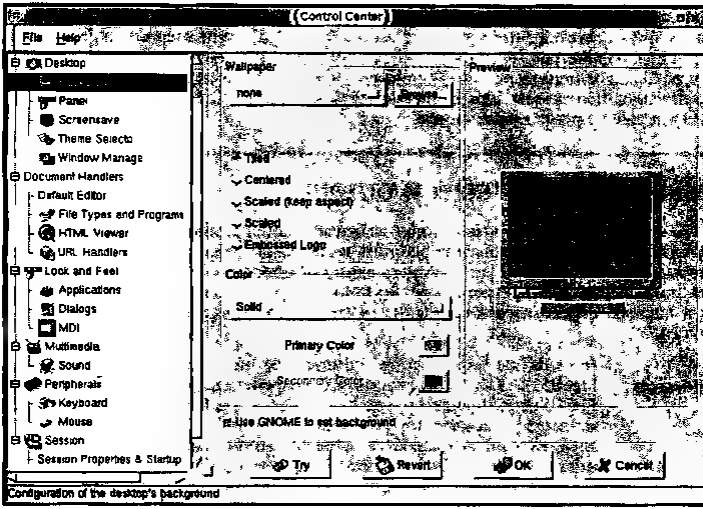


Рис 3 5 Программа Control Center позволяет настраивать параметры оболочки GNOME и связанных с ней программ

В левой части окна программы перечислены группы установок. На рис. 3.5 выбран модуль Desktop⇨Background, в котором задается цвет фона и выбирается фоновый ри сунок Ниже перечислены доступные категории

- **Рабочий стол (Desktop)** В этой категории можно задать цвет или рисунок фо на настроить панель задач и хранитель экрана, загрузить тему (группу согласованных установок графической среды) и поменять диспетчер окон.
- **Обработчики документов (Document Handlers)** Можно указать программе GNOME необходимый текстовый редактор, типы файлов связанные с программами, способ отображения HTML-документов и программы для обработки конкретных типов URL-адресов
- **Внешний вид (Look and Feel)** Модули данной категории позволяют задавать параметры пользовательского интерфейса оболочки GNOME. Они не относят ся к большинству программ, которые не написаны специально для этой оболочки так что не удивляйтесь, если изменения затронут не все окна
- **Мультимедиа (Multimedia)** Разрешается настраивать некоторые установки зву ковой платы и закреплять звуковые файлы за определенными событиями та кими как вход в систему выход из нее, получение электронной почты и т д
- **Периферийные устройства (Peripherals)**. Программа позволяет регулировать скорость повтора клавиш и перемещения курсора мыши, менять функции кнопок мыши и т.п
- **Сеанс (Session)** С помощью установок группы Session можно указывать, ка кие программы следует запускать при регистрации в системе и должна ли программа GNOME спрашивать подтверждение перед выходом из системы

Не забывайте о том что это пользовательские установки Они будут разными для разных пользователей, чтобы каждый мог работать за компьютером так, как ему хочется Ряд параметров, влияющих на работу всей системы с помощью программы Control Center изменить нельзя Например, она не позволяет загружать драйверы инсталлировать программы или форматировать диски Это административные задачи описываемые в других главах книги

Выход из системы и останов

По окончании работы можно либо выйти из системы либо остановить ее. В первом случае компьютер продолжит работать в текстовом режиме, включая все серверы, которые были запущены. Система лишь отключает доступ к учетной записи пользователя, требуя повторной регистрации Во втором случае останавливаются все программы и система переводится в состояние, в котором компьютер можно безопасно выключить

Никогда не выключайте компьютер, не остановив предварительно FreeBSD! Как и большинство современных операционных систем, FreeBSD выполняет много операций в фоновом режиме и *кэширует* данные, записываемые на диск. Это означает, что перед записью на диск данные сохраняются в резидентном кэш-буфере, с тем чтобы можно было объединить несколько операций записи в одну для повышения производительности. Таким образом, некорректное выключение компьютера может привести к потере данных и повреждению файловых систем

Выход из системы в текстовом и графическом режимах

Если вы на какое-то время покидаете свое рабочее место, выйдите из системы чтобы в ваше отсутствие никто не смог скомпрометировать вашу учетную запись, например послать письмо по электронной почте от вашего имени и попытаться взломать другие компьютеры При выходе из системы уничтожаются все запущенные программы, включая те, что работают в фоновом режиме

Если работа велась в текстовом режиме, достаточно ввести команду `logout` Система завершит сеанс работы с учетной записью, и на экране снова появится приглашение `login:` Если вы регистрировались в системе несколько раз, не забудьте завершить все сеансы работы

По умолчанию при выходе из системы экран не очищается Если на экране отображались важные данные, введите сначала `clean`, а потом `logout`

В графической среде команда `logout`, введенная в окне терминала, не работает поскольку терминальные окна не являются сеансами регистрации в системе Вместо этого нужно остановить X-сервер или завершить X-сеанс, выбрав соответствующую команду в стартовом меню. Как именно она называется, зависит от используемой графической оболочки В среде GNOME это команда Log Out. В некоторых автономных диспетчерах окон подобная команда находится в контекстном меню которое вызывается щелчком правой кнопки мыши на рабочем столе.

Далее появится регистрационный экран XDM (если в системе используется программа XDM) либо исходный экран текстового режима с большим числом сообще

ний, выданных X сервером (их можно проигнорировать если X сеанс завершился корректно) Теперь введите `logout`, чтобы выйти из системы

На заметку В некоторых случаях X сеанс не завершается корректно при выборе команды типа Log Out В этом случае нужно нажать <Ctrl+Alt+Backspace>

Останов системы в текстовом и графическом режимах

Останов системы позволяет подготовить компьютер к безопасному выключению питания. Для останова системы нужно быть суперпользователем. Соответствующая команда называется `shutdown`. Ее можно ввести в текстовом сеансе регистрации или в терминальном окне графической среды. Синтаксис команды таков:

```
shutdown [опции] время [предупреждающее_сообщение]
```

Аргумент *время* определяет, когда должна быть выключена система Поддерживаются три спецификации.

- **now** Ключевое слово now означает немедленный останов Его удобно использовать в однопользовательской рабочей станции.
- *+число* Эта спецификация позволяет запланировать останов системы через указанное число минут. Например выражение +30 означает останов через полчаса.
- *ггммдччмм* Чтобы запланировать останов на конкретное время, воспользуйтесь данной спецификацией, в которой первые две цифры обозначают год следующие две — месяц, затем, по очереди, — день час и минуту (номер часа задается в 24-часовом формате)

Необязательный аргумент *предупреждающее_сообщение* задает строку отображаемую на текстовых консолях всех зарегистрированных пользователей как минимум за десять часов до запланированного останова Например, сообщение вида "`Shut down for disk upgrade; up at 7:00 AM`" информирует пользователей о том, какова причина выключения сисемы и когда она снова станет доступной.

Команда `shutdown` поддерживает также ряд дополнительных опций

- `-h` После останова система полностью прекратит работу. Далее нужно вручную выключить питание компьютера
- `-p` После останова питание компьютера будет выключено автоматическн. Необходимо, чтобы компьютер поддерживал возможность программируемого выключения питания, что характерно для большинства современных компьютеров
- `-r` После останова произойдет перезагрузка системы
- `k`. При наличии этой опции все пользователи будут принудительно выведены из системы и не смогут вновь зарегистрироваться в ней, хотя сама система продолжит работу. Это может понадобиться, если администратор запланировал инсталляцию нового программного обеспечения или проверку в безопасной среде
- `o` В режимах `-h`, `-p` и `-r` подразумевается, что команда `shutdown` передает управление демону `init`, который выполняет всю необходимую работу. При наличии опции `-o` команда `shutdown` все делает сама, хотя обычно это не требуется.
- `n` Эта опция может использоваться совместно с предыдущей и заставляет команду `shutdown` пропустить этап очистки кэш-буферов файловых систем Та

кое поведение команды способно привести к повреждению файловых систем и потому применять опцию не рекомендуется.

- - (дефис). Если в качестве опции указан дефис, команда `shutdown` предложит ввести предупреждающее сообщение вместо того которое указано в командной строке.

После ввода команды `shutdown` FreeBSD выполняет в обратном порядке действия предпринятые в процессе загрузки системы Сначала завершаются все работающие программы, включая X-сервер и все остальные серверы. Затем происходит демонтирование сетевых файловых систем и дисковых разделов. Далее все зависит от опций команды. Если указана опция `-h`, появится сообщение о том, что система остановлена; можно выключить питание компьютера или нажать кнопку сброса, чтобы перезагрузиться. Если указана опция `-p` или `-r`, произойдет либо выключение питания (если компьютер поддерживает возможность программного выключения), либо перезагрузка.

В качестве альтернативы команде `shutdown` можно нажать <Ctrl+Alt+Del>. В результате произойдет останов системы с последующей перезагрузкой. Это проще, чем запоминать различные опции команды `shutdown`, и удобнее для неопытных пользователей. Естественно, если планируется выключить компьютер, то перед началом перезагрузки нужно нажать кнопку выключения питания. Некоторые графические программы регистрации, предназначенные для замены XDM, предлагают пользователям возможность выключить компьютер. Для этого предоставляется соответствующая кнопка или команда меню. Такая возможность есть в GDM и KDM (являются компонентами оболочек GNOME и KDE соответственно)

Когда нужно остенавливвть систему

В списках рассылки и телеконференциях периодическн ведутся дебаты о том ко гда следует выключать питание компьютера Существуют два принципиальных подхода: никогда не выключать и выключать регулярно в целях экономии электроэнергии.

Сторонники первого подхода считают, что аппаратные сбои чаще всего происходят из-за перегрузки, связанной с включением и выключением устройств Изменения температуры и напряжения приводят к большему износу устройства, чем непрерывная работа. Рассмотрим обычную электрическую лампочку Как правило, она перегорает когда включается свет, а не от длительной работы. То же самое справедливо и в отношении компьютерных компонентов, поэтому срок службы компьютера можно продлить, если как можно реже выключать его.

Сторонники противоположиого подхода придерживаются мнения, что увеличение срока службы компьютера, достигаемое за счет непрерывной, безостановочной работы, невелико, особенно если учесть, что наиболее важные компоненты устареют задолго до того, как выйдут из строя. В то же время постоянно включенный компьютер потребляет электроэнергию, и экономия электроэнергии в этом случае важнее

Оба подхода имеют свои преимущества, и выводы читатели могут сделать сами. Не следует также забывать о том, что во FreeBSD есть средства автоматического обслуживания. Определенные программы запускаются поздно ночью и выполняют служебные операции, например удаляют старые файлы из каталога `/tmp`, как описывается в главе 28. Если постоянно выключать компьютер на ночь придется либо перенести эти процедуры на другое время, либо иногда оставлять компьютер включениым всю ночь

Естественно, все эти рассуждения относятся к рабочим станциям. Большинство серверов работает 24 часа в сутки 7 дней в неделю Их выключают только для замены аппаратных компонентов, ремонта и т п

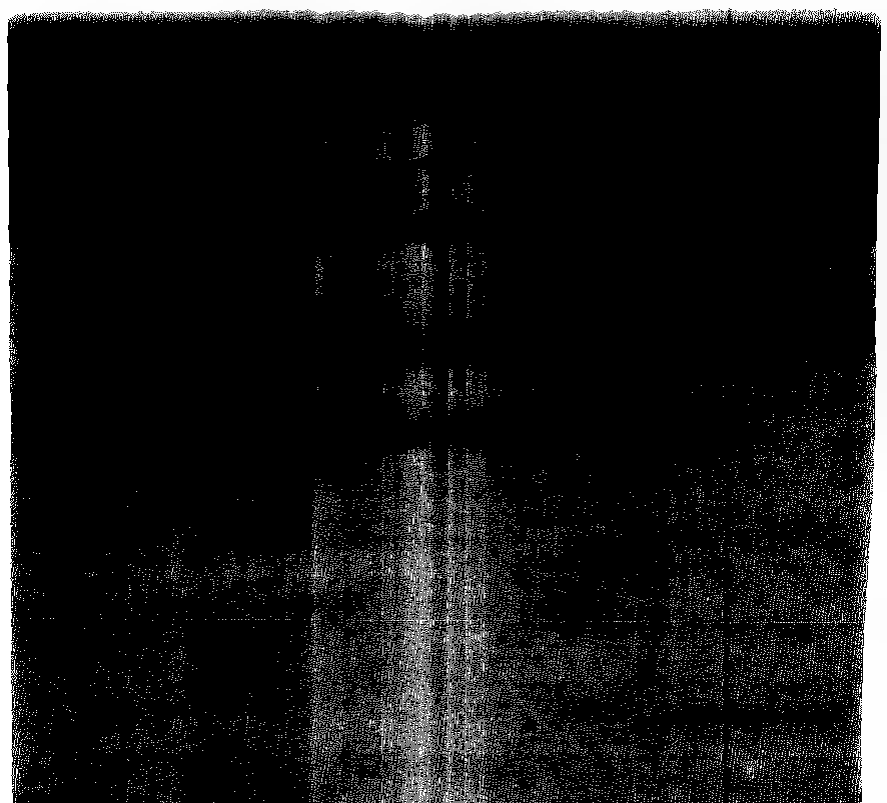
Резюме

Отличия FreeBSD от Windows начинают проявляться почти сразу, когда включается питание компьютера. В процессе загрузки системы выдается множество сообщений о выполняемых действиях. Далее по умолчанию появляется экран регистрации в текстовом режиме. После регистрации в системе можно запускать текстовые программы. Во FreeBSD поддерживается множество команд UNIX, предназначенных для просмотра файлов и манипулирования ими. Важно уметь пользоваться хотя бы некоторыми из этих команд.

Для запуска графических программ нужно предварительно запустить X-сервер, введя команду `startx`. Эта команда загрузит также графическую оболочку, выбранную по умолчанию или сконфигурированную пользователем. Некоторые графические оболочки FreeBSD напоминают интерфейс Windows и других современных операционных систем, поэтому у пользователей этих систем не должно возникнуть проблем с адаптацией к FreeBSD.

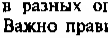
По окончании работы нужно выйти из системы, чтобы свести к минимуму риск взлома учетной записи. Если требуется выключить компьютер или перезагрузить систему, старайтесь делать это по всем правилам, а не просто нажимать кнопки выключения или сброса.

Полный справочник по



Как описывалось в предыдущих главах, в частности в главе 2, FreeBSD может быть единственной операционной системой компьютера, а может работать совместно с другими операционными системами. Во втором случае важно понимать, как FreeBSD сосуществует со своими "коллегами". Это позволит максимально эффективно использовать ресурсы компьютера и избежать губительных ошибок (можно по неосторожности полностью уничтожить инсталляцию "соседней" ОС).

В этой главе рассматриваются особенности сосуществования FreeBSD с другими операционными системами. Сначала описывается процесс начальной загрузки с акцентом на конфигурирование и применение различных системных загрузчиков. Следующая тема — управление разделами. Каждый раздел содержит файловую систему, а в разных операционных системах используются файловые системы разных типов. Важно правильно организовать доступ к разделам "соседних" операционных систем. В конце главы рассказывается о том, как запускать во FreeBSD программы, написанные для других операционных систем, в частности приложения Linux и Windows.



Читателям, у которых на компьютере установлена только FreeBSD, большая часть главы не представляет интереса. Полезной будет информация о файловых системах, если предполагается монтировать гибкие диски, Zip-диски и т. п.

Настройка процесса начальной загрузки

В главе 2 приводилась базовая информация об инсталляции загрузчика FreeBSD. Этот загрузчик хорошо работает в простых конфигурациях, когда на компьютере установлена только FreeBSD или на одном жестком диске установлены FreeBSD и Windows. В более сложных конфигурациях может понадобиться модифицировать стандартную конфигурацию загрузчика или даже воспользоваться другим загрузчиком. В этом разделе рассказывается о различных загрузчиках и о том, как с помощью загрузчика FreeBSD передавать специальные параметры ядру системы. Эти параметры позволяют загружаться с устройств, вызывающих проблемы в случае стандартной конфигурации.

Системные загрузчики

Когда компьютер семейства x86 загружается, он ищет код для запуска системы в нескольких местах. В первую очередь происходит обращение к ПЗУ (постоянное запоминающее устройство), где хранится BIOS (Basic Input/Output System — базовая система ввода-вывода) BIOS, в свою очередь, обращается к *загрузочному сектору* одного или нескольких загрузочных устройств. Обычно система загружается с жесткого диска, либо потому, что он проверяется первым, либо потому, что другие кандидаты в момент опроса не содержат загрузочного носителя. Но в процессе инсталляции системы можно переконфигурировать BIOS, сделав загрузочным устройством дисковод гибких дисков или дисковод CD-ROM. Загрузочный сектор жесткого диска называют *главной загрузочной записью* (ГЗЗ). Стандартная ГЗЗ содержит код, который в типичной среде DOS или Windows перенаправляет процесс начальной загрузки к загрузочному сектору корневого раздела. Код, содержащийся в корневом разделе, осуществляет загрузку DOS или Windows. Простейшая конфигурация, в которой загружается только одна операционная система, изображена на рис. 4.1 (раздел FreeBSD присутствует, но не участвует в процессе начальной загрузки).

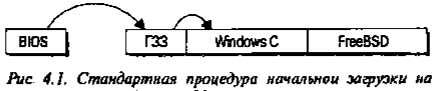


Рис. 4.1. Стандартная процедура начальной загрузки на компьютере семейства x86

В случае мультисистемного компьютера на одном из этапов этого процесса должна появиться возможность выбора операционной системы. Для FreeBSD традиционный подход заключается в замене кода ГЗЗ новым кодом, позволяющим сделать выбор между FreeBSD и другими операционными системами, установленными на компьютере. Но такой подход не всегда работает. Некоторые операционные системы поддерживают собственные загрузчики. Существуют также сторонние загрузчики, которые могут заменять код ГЗЗ собственным кодом, менять содержимое загрузочного сектора Windows, вмешиваться в процесс начальной загрузки на последующих этапах или инсталлировать специальный раздел, предназначенный для изменения хода начальной загрузки. Подробнее об этих загрузчиках рассказывается на с. 113.

Важным фактором является так называемое *1024-цилиндровое ограничение*. Традиционно в компьютерах семейства x86 применяется *схема адресации "цилиндр—головка—сектор"* (ЦГС), в которой каждый сектор жесткого диска уникальным образом идентифицируется тремя числами. Самый первый код BIOS был написан в то время, когда жесткие диски имели гораздо меньшую емкость, чем сегодня, поэтому на адреса ЦГС налагались ограничения, из-за которых максимальная емкость жесткого диска составляет около 8 Гбайт (ограничения старого интерфейса IDE снижали этот показатель до 504 Мбайт, однако в середине 90-х код BIOS подвергся модификации, которая позволила достичь верхнего предела — 8 Гбайт). Процесс начальной загрузки во многом зависит от BIOS, следовательно, упомянутые ограничения актуальны для любой операционной системы.

В конце 90-х код BIOS стал включать расширения (называемые *расширенными вызовами INT13*), позволяющие преодолеть 1024-цилиндровое ограничение. Впрочем, необходимо, чтобы системные загрузчики тоже поддерживали эти расширения. Загрузчик FreeBSD включает такую поддержку, но для загрузки файлов FreeBSD, находящихся после 1024-го цилиндра, придется установить специальные параметры (емкость диска может быть больше, чем 1024 цилиндра, однако специальные опции не нужны, если загрузочные файлы расположены в пределах первых 1024 цилиндров). Сторонним загрузчикам тоже нужна поддержка расширенных вызовов INT13, чтобы иметь возможность загружать FreeBSD (или любую другую операционную систему) из "дальних" областей диска.

Использование загрузчика FreeBSD

По умолчанию загрузчик FreeBSD позволяет запускать саму FreeBSD и, возможно, еще одну операционную систему, такую как DOS или Windows. Если же необходимо после включения компьютера отобразить меню выбора операционной системы или внести другие изменения в конфигурацию загрузчика, воспользуйтесь командой `boot0cfg`. Синтаксис ее вызова таков:

```
boot0cfg [-Bv] [-b загрузочный_файл] [-d диск] [ f файл] [-m маска] ↵
[-o флаги] [-s первичный_раздел] [-t такты] диск
```

В большинстве случаев достаточно указать одну-две опции. Например, если ввести **`boot0cfg -B /dev/ad0`** от имени суперпользователя, то загрузчик будет инсталлирован на первый жесткий диск EIDE. Он (загрузчик) будет просматривать все загрузочные разделы и предлагать пользователю возможность выбора варианта загрузки (если

в системе установлены диски SCSI поменяйте /dev/ad0 на /dev/da0) Ниже описаны опции команды.

- **B.** Эта опция заставляет команду инсталлировать код ГЗЗ, позволяющий выбрать операционную систему Опустите эту опцию, если используется сторонний первичный загрузчик.
- **v** Обычно команда boot0cfg отображает минимум сообщений Эта опция заставляет команду выдавать дополнительную информацию, например указывать, какие первичные разделы делаются загрузочными.
- **b** *загрузочный_файл*. Чтобы инсталлировать загрузчик команда boot0cfg копирует заданный файл в ГЗЗ. По умолчанию это файл /boot/boot0, но можно задать другой файл если планируется использовать иной загрузчик, например экспериментальный загрузчик FreeBSD (стороннему загрузчику нужно позволить самостоятельно записать себя в ГЗЗ)
- **d** *диск*. BIOS ссылается на жесткие диски по номерам Первый физический диск обычно имеет номер 0x80 (т.е. шестнадцатеричное 80 или десятичное 128), второй — 0x81 и т.д. Если BIOS сконфигурирована на загрузку со второго или последующего дисков, то необходимо воспользоваться опцией **-d** *диск*, где в качестве аргумента должен быть задан номер диска. Для простоты упорядочьте диски так, чтобы загрузчик FreeBSD и операционная система которая должна загружаться с первого физического диска, находились на этом диске Тогда и опция **d** не понадобится
- **-f** *файл* Эта опция заставляет команду записать образ существующей ГЗЗ в указанный файл Это полезно делать, т к в случае возникновения проблем исходную ГЗЗ можно восстановить с помощью утилиты dd.
- **-m** *маска* Обычно команда boot0cfg сама определяет, какие первичные разделы должны быть указаны в начальном загрузочном меню. Если же ее "догадки" неверны, то с помощью опции **-m** можно переопределить их. Аргумент *маска* принимает значения от 0 (выбор первичных разделов не разрешен) до 0xf (десятичное 15, все первичные разделы доступны). Он представляет собой четырехбитовый код, каждый бит которого соответствует одиому разделу.
- **o** *флаги* С помощью этой опции можно задать три дополнительных флага. К каждому флагу можно добавить префикс no чтобы изменить его значение на противоположное.
  - **packet**. По умолчанию установлен флаг nopacket, что заставляет первичный загрузчик использовать старые режимы адресации ЦГС В результате FreeBSD сможет загрузиться только из первых 8 Гбайт диска Если установить флаг packet, FreeBSD будет использовать расширенные вызовы INT13, что позволит преодолеть 1024-цилиндровое ограничение
  - **setdrv** Установите этот флаг, если были изменены дисковые привязки в BIOS и при этом использовалась опция **d** *диск*, иначе FreeBSD может не загрузиться корректно
  - **update** Обычно FreeBSD меняет стандартный загрузочный раздел всякий раз, когда пользователь выбирает другую операционную систему для загрузки В следующий раз выбранная система будет предложена по умолчанию Если установлен флаг noupdate, загрузчик не станет вносить изменения в ГЗЗ и стандартный выбор будет всегда одинаков.
- **s** *первичный_раздел*. Эта опция обычно используется совместно с опцией o noupdate и задает номер первичного раздела, выбираемого загрузчиком по умолчанию Аргумент *первичный_раздел* принимает значения от 1 до 4

- `t` *такты* Обычно система делает паузу в несколько секунд чтобы дать пользователю возможность выбрать загружаемую операционную систему С помощью этой опции можно изменить длительность паузы в тактах (1 секунда это 18,2 такта)
- *диск*. Этот аргумент задает диск, ГЗЗ которого модифицируется командой `boot0cfg`. Для дисков EIDE это, как правило устройство `/dev/ad0`, а для дисков SCSI — устройство `/dev/da0`

Необходимость в команде `boot0cfg` может возникнуть по нескольким причинам Одна из них — изменение установок по умолчанию, например добавление новой операционной системы в список выбора Другая - переинсталляция загрузчика FreeBSD после того, как он был затерт другой операционной системой Разумеется, в этом случае придется загрузить FreeBSD с аварийной дискеты.

Операционные системы компании Microsoft в процессе инсталляции обычно восстанавливают стандартный вид ГЗЗ Таким образом, если Windows устанавливается после FreeBSD, загрузчик FreeBSD придется переинсталлировать

Одно из основных ограничений загрузчика FreeBSD заключается в том, что он способен перенаправлять процесс начальной загрузки только к операционным системам, находящимся в первичных разделах x86 Но есть операционные системы, например Linux и OS/2, которые могут загружаться из логических разделов. Если такая система инсталлирована в логическом разделе, необходимо либо использовать другой первичный загрузчик, либо установить в другой первичный раздел сторонний загрузчик, который будет загружать операционную систему из логического раздела

Задание параметров ядра

После того как выполнился код, хранящийся в ГЗЗ, управление передается другому коду. В случае загрузчика FreeBSD это называется *этапом 1* и *этапом 2* начальной загрузки (выполнение кода ГЗЗ — *этап 0*) Во FreeBSD на этих этапах можно передавать ядру различные конфигурационные параметры

Если нужно переинсталлировать загрузчики этапов 1 и 2, введите команду **`disklabel /dev/`***`имя`*, где аргумент *имя* задает первичный раздел FreeBSD, например `/dev/ad0s2`. Это может понадобиться, если диск был поврежден

Для человека, сидящего за компьютером этот этап начальной загрузки начинается с приглашения вида

```
Hit [Enter]  to boot immediately, or any other key for command prompt
Booting [/boot/kernel/kernel] in 9 seconds...
```

Во второй строке счетчик времени уменьшается до нуля Если до этого успеть нажать какую-нибудь клавишу, кроме <Enter>, появится приглашение командной строки, в которой можно задать параметры загрузки. Ниже перечислены доступные команды

- **`autoboot`** *секунды* — начинает отсчет времени заново задавая новый временной интервал
- **`boot [-`***опции***`] [`***ядро***`]`** — позволяет загрузить указанное ядро и передать ему требуемые параметры.
- **`load [-t`** *тип***`]`** *модуль* — загружает указанный модуль Модуль ядра — это драйвер или расширение ядра С помощью модулей включается поддержка не

стандартного оборудования. Это, впрочем, можно сделать и по окончании загрузки, так что данная команда нужна только в том случае, когда нестандартное устройство, например дисковый контроллер, само по себе обеспечивает загрузку системы.

- `ls [ l] [путь]` — это аналог стандартной команды `ls`. Выводит список файлов заданного каталога. На ранних стадиях начальной загрузки не все разделы могут быть доступны.
- `lsdev [-v]` — выводит список устройств, с которых можно загружать модули. При наличии опции `-v` выдается более подробный отчет.
- `lsmod [-v]` — выводит список загруженных модулей. Как и в случае команды `lsdev`, опция `-v` позволяет получить более подробный отчет.
- `more` *имя файла* — отображает содержимое заданного файла, делая паузу при заполнении экрана. Это удобно при чтении документации к модулям.
- `reboot` — перезагружает систему.
- `unload` — выгружает все загруженные ранее модули.

На этом этапе можно загрузить другое ядро, например в целях тестирования. Может также понадобиться загрузить или, наоборот, выгрузить определенные модули ядра для обеспечения загрузки системы или передать ядру дополнительные параметры (о них будет рассказываться в главе 12). Важно понимать, что вышеперечисленные команды доступны независимо от того, находится ли в ГЗЗ загрузчик FreeBSD или сторонний загрузчик.

Использование стороннего загрузчика

На сегодняшний день существует множество системных загрузчиков. С большинством операционных систем поставляются загрузчики, способные взаимодействовать с несколькими системами, включая FreeBSD. Но необязательно использовать именно их. Есть загрузчики, не зависящие от операционной системы. Перечислим наиболее распространенные загрузчики.

- **LILO (Linux Loader)** — традиционный загрузчик Linux. Он может находиться в ГЗЗ, в корневом разделе Linux или на дискете. LILO — очень гибкий загрузчик, но эта гибкость достигается путем утомительного конфигурирования.
- **GRUB (Grand Unified Bootloader)** — в настоящее время входит в состав многих дистрибутивов Linux. Его можно сконфигурировать для загрузки FreeBSD.
- **Windows OS Loader.** Программа OS Loader (называется также NTLDR) является загрузчиком Windows NT/2000/XP. Ей не хватает гибкости, а конфигурационные файлы могут храниться только в разделах Windows. В качестве первичного загрузчика почти всегда лучше использовать загрузчик FreeBSD. Программа OS Loader может быть вторичным загрузчиком, запускаемым после выбора Windows в меню загрузчика FreeBSD.
- **OS/2 Boot Manager** — довольно гибкий загрузчик, занимающий собственный выделенный раздел. Программа может загружать FreeBSD, OS/2, Linux и многие другие операционные системы.
- **System Commander** — очень гибкий загрузчик, выпущенный компанией V-Com (`www.v-com.com`). Он записывается в ГЗЗ и может загружать практически любую операционную систему, включая FreeBSD, из первичного или (если операционная система это поддерживает) логического раздела x86. Он также умеет скрывать первичные разделы от других операционных систем, таких как Windows.

- **BootMagic** — поставляется с программой PartitionMagic компании PowerQuest (`www.powerquest.com`), но доступен и отдельно от нее. Это довольно мощный загрузчик, записываемый в загрузочный сектор раздела Windows. Он может загружать большинство операционных систем, включая FreeBSD, из первичного или (если операционная система это поддерживает) логического раздела x86.

Как правило, сторонний загрузчик записывается в ГЗЗ, замещая загрузчик FreeBSD. Если выбрать для загрузки FreeBSD, будет запущен загрузчик этапа 1 из первичного раздела FreeBSD, который выдаст приглашение, показанное в предыдущем подразделе.

В большинстве загрузчиков доступные операционные системы задаются с помощью меню. Отдельного упоминания заслуживает LILO. Если используется этот загрузчик, необходимо добавить в файл `/etc/lilo.conf` запись примерно такого вида:

```
other = /dev/hda2
    label = freebsd
    table = /dev/hda
```

Эта запись говорит о том, что FreeBSD загружается с Linux-устройства `/dev/hda2` (аналог `/dev/ad0s2` во FreeBSD), а таблица разделов хранится на устройстве `/dev/hda` (`/dev/ad0` во FreeBSD). Более подробную информацию о конфигурировании LILO можно найти на man-странице файла `lilo.conf`. После того как запись создана, выполните команду `lilo` от имени суперпользователя, чтобы переинсталлировать LILO. Теперь, чтобы загрузить FreeBSD, нужно ввести **`freebsd`** в строке приглашения LILO либо выбрать FreeBSD из меню загрузчика, если LILO сконфигурирован на отображение такого меню.

Управление разделами FreeBSD и x86

Разделы — это базовые элементы управления дисками. Считается, что каждая операционная система, установленная на компьютере, должна занимать один или несколько разделов, не пересекаясь с другими операционными системами. Важно так все сконфигурировать, чтобы "соседи" ненароком не повредили разделы FreeBSD. С другой стороны, часто создается раздел общего доступа, через который происходит обмен файлами между системами. Это противоречит исходной цели (обособить операционные системы), но если правильно настроить разделы, риск их повреждения будет минимальным.

Типы разделов

В главе 1 рассказывалось о том, что такое раздел и какие существуют типы разделов, а в главе 2 описывались программы, используемые во FreeBSD для управления разделами. Напомним, что компьютер семейства x86, работающий под управлением FreeBSD, может содержать несколько типов разделов:

- **Первичный раздел**. Исходным типом раздела в компьютерах семейства x86 является *первичный раздел*. Система разбивки на разделы поддерживает максимум четыре таких раздела. Многие операционные системы, включая FreeBSD, должны загружаться именно из первичного раздела.
- **Расширенный раздел**. Поскольку четырех первичных разделов часто бывает недостаточно, был придуман специальный заменитель, называемый *расширенным разделом*. Такой раздел занимает одну из четырех позиций, отведенных для первичных разделов, но может содержать множество других разделов.

- **Логический раздел.** *Логические разделы* находятся в пределах расширенного раздела. В Windows таким разделам присваиваются буквенные обозначения от D: и далее, в зависимости от конфигурации. Большинство операционных систем невозможно загрузить из логического раздела, но есть и исключения. Во FreeBSD логические разделы по умолчанию не создаются, хотя можно сконфигурировать систему для доступа к ним.
- **Разделы FreeBSD.** FreeBSD обычно содержится в одном первичном разделе, который разбивается на несколько разделов FreeBSD, являющихся эквивалентами логических разделов в обычной схеме.

Важно понимать разницу между типами разделов. Компьютеры, на которых установлена только Windows, имеют обычно один первичный раздел. Если же помимо Windows инсталлируется FreeBSD, то создается еще и первичный раздел FreeBSD, который разбивается на подразделы.

Защита разделов FreeBSD от повреждения другими операционными системами

Программа разбивки на разделы назначает каждому разделу *код типа*. Некоторые операционные системы, в частности DOS, Windows и OS/2, проверяют эти коды, чтобы избежать повреждения "чужих" разделов. Таким образом, чтобы защитить первичный раздел FreeBSD, нужно пометить его соответствующим кодом. При инсталляции FreeBSD это делается автоматически, так что беспокоиться не приходится. Но некоторые утилиты обслуживания дисков могут менять коды разделов, если неосторожно ими пользоваться.

Наиболее распространенные коды перечислены в табл. 4.1. Для дисков, емкость которых превосходит 1024-цилиндровое ограничение, Windows использует коды, связанные с режимом LBA. Старые операционные системы могут не распознать такие разделы.

Таблица 4.1. Коды разделов и связанные с ними файловые системы

Шестнадцатеричный код	*Десятичный код*	*Файловая система*	*Операционные системы*
0x01	1	FAT-12	DOS, большинство других систем
0x04	4	FAT 6 (< 32 Мбайт)	DOS, большинство других систем
0x05	5	Расширенный раздел	Все операционные системы персональных компьютеров
0x06	6	FAT 16	DOS, Windows, большинство других систем
0x07	7	HPFS или NTFS	OS/2, Windows NT/2000/XP
0x0a	0	OS/2 Boot Manager	OS/2
0x0b	11	FAT-32	Windows 95 OSR2 и выше
0x0c	12	FAT-32 (режим LBA)	Windows 95 OSR2 и выше
0x0a	14	FAT-16 (режим LBA)	Windows 95 OSR2 и выше
0x0f	15	расширенный (режим LBA)	Windows 95 OSR2 и выше
0x82	130	Раздел подкачки Linux или Solaris	Linux или Solaris

Шестнадцатеричный код	*Десятичный код*	*Файловая система*	*Операционные системы*
0x83	131	Файловые системы Linux (ext2fs XFS, ReiserFS и др.)	Linux
0xa5	165	Первичный раздел BSD	BSD-системы, включая FreeBSD
0xeb	235	BeFS	BeOS

На заметку Таблица 4.1 не является полной. Многие менее распространенные операционные системы используют собственные коды разделов. Кроме того, даже популярные операционные системы распознают дополнительные коды специального назначения.

Узнать типы разделов жесткого диска позволяет команда `fdisk`. Укажите опцию `s` и идентификатор диска, чтобы просмотреть таблицу первичного раздела:

```
# fdisk -s /dev/ad0
/dev/ad0: 784 cyl 255 hd 63 sec
Part          Start        Size Type Flags
   1             63     4192902 0x0b 0x00
   2        4192965     8401995 0x0b 0x80
```

Эта команда сообщает размер диска (в виде количества цилиндров, головок и секторов), размер каждого раздела (в виде количества 512-байтовых секторов) и его код. Если раздел FreeBSD имеет неправильный код, то, скорее всего, другая операционная система пыталась получить к нему доступ и повредила. Чтобы исправить ситуацию, воспользуйтесь опцией `-u` команды `fdisk`. Команда предложит обновить каждый элемент информации о диске. В большинстве запросов нужно нажать <Enter>; поменяйте лишь код раздела.

Например, следующий диалог демонстрирует изменение кода первого раздела из предыдущего примера.

```
The data for partition 1 is

sysid 11, (DOS or Windows 95 with 32 bit FAT)
    start 63, size 4192902 (2047 Meg., flag 0
        beg: cyl 0/ head 1/ sector 1;
        end: cyl 260/ head 254/ sector 63
Do you want to change it? [n] y
Supply a decimal value for "sysid (165=FreeBSD    [11] 165
```

На заметку Несмотря на то что команда `fdisk -s` сообщает шестнадцатеричный код раздела, команда `fdisk -u` отображает в приглашении десятичный код и ожидает ввода именно десятичного кода. С этой целью в табл. 4.1 были приведены оба варианта.

Некоторые операционные системы, например Linux, не проверяют коды разделов, задавая их только в процессе инсталляции. Эти системы нужно сконфигурировать так, чтобы они игнорировали первичный раздел FreeBSD. Большинство дистрибутивов Linux делает это по умолчанию, так что в Linux проблем не возникает. Если все же нужно получить доступ к разделам FreeBSD из Linux, включите в Linux специальный режим доступа (с. 125). Доступ к разделам FreeBSD из других BSD систем, в частности OpenBSD и NetBSD, осуществляется довольно просто. Как правило, каждая BSD система занимает отдельный первичный раздел, и при монтировании каталогов другой BSD-системы нужно указать лишь раздел.

Доступ к внешним разделам из FreeBSD

Во FreeBSD каждый раздел имеет уникальный идентификатор в каталоге /dev Имена файлов этого каталога состоят из нескольких компонентов

- **Тип устройства.** Жесткие диски помечаются двухбуквенным кодом Например для EIDE-дисков используется код ad, для SCSI-дисков — код da USB-диски интерпретируются как SCSI-устройства, поэтому тоже имеют код da
- **Номер устройства** У каждого физического диска есть номер. Первый диск определенного типа обычно имеет номер 0, второй — 1 и т.д. Для EIDE-дисков нумерация зависит от контроллера и позиции диска в цепочке. Главный диск первичного контроллера имеет номер 0, подчиненный диск первичного контроллера — номер 1, главный диск вторичного контроллера — номер 2 и т.д. SCSI-диски нумеруются последовательно, начиная с нуля.
- **Номер первичного раздела** Первичные разделы нумеруются от 1 до 4. Этому номеру предшествует буква s. Нумерация обычно начинается с единицы, но некоторые дисковые утилиты создают первичные разделы в особом порядке и позволяют удалять разделы, вследствие чего возникают разрывы в нумерации Логические разделы интерпретируются подобно первичным, но их нумерация начинается с 5 Таким образом, если диск содержит два первичных и два логических раздела, их нумерация будет такой. 1, 2, 5 и 6.
- **Буква, обозначающая раздел.** После номера первичного раздела указывается буква обозначающая подраздел FreeBSD в рамках первичного раздела Эти буквы интерпретируются особым образом (табл 4.2)

Таблица 4 2 Буквенные обозначения разделов FreeBSD

Буква	*Интерпретация*
a	Корневой раздел (/)
b	Область подкачки
c	Весь первичный раздел (использовать такое обозначение не рекомендуется)
d и далее по алфавиту	Разделы общего назначения

При ссылке на первичные или логические разделы, не относящиеся к FreeBSD нужно опускать суффикс раздела, а при доступе ко всему диску (с помощью утилиты fdisk, например) не нужно указывать номер первичного раздела В табл 4 3 приведены идентификаторы разделов и объясняется их назначение

Таблица 4.3. Примеры идентификаторов разделов

Раздел FreeBSD	*Назначение*
/dev/ad0	Первый диск EIDE
/dev/da0	Первый диск SCSI
/dev/ad0s1	Первый раздел первого диска EIDE
/dev/da.s3a	Корневой раздел FreeBSD находящийся в третьем первичном разделе второго диска SCSI
/dev/ad2s6	В орой логический раздел главного диска вторичного контроллера EIDE

Тем, кто не знаком с обозначениями разделов FreeBSD, подобные соглашения могут показаться странными, но к ним быстро привыкаешь В файле /etc/fstab перечислены разделы монтируемые при загрузке FreeBSD Подробнее этот файл рассмотрен в главе 7

На заметку Пользователи, имеющие опыт работы в Linux должны были заметить что идентификаторы дисков Linux отличаются от аналогичных идентификаторов FreeBSD. Действительно, в этих операционных системах используются разные правила именования дисковых устройств

Помимо дисковых разделов, создаются идентификаторы и других дисковых устройств К примеру, устройство `/dev/fd0` обозначает дисковод гибких дисков, устройство `/dev/acd0` (как вариант — `/dev/acd0a` или `/dev/acd0c`) представляет EIDE дисковод CD-ROM, а устройство `/dev/cd0a` — это SCSI-дисковод CD-ROM Если в системе несколько дисковых устройств одного типа (например, два дисковода гибких дисков), номера дополнительных устройств будут увеличиваться на единицу (`/dev/fd1`) Дискеты и компакт-диски обычно не разбиваются на разделы хотя некоторые идентификаторы подразумевают такую возможность.

Zip, LS-120, магнитооптические и другие дисковые устройства со съемными носителями интерпретируются подобно жестким дискам. Zip-диски обычно имеют один первичный раздел номер 4 с таблицей разделов К примеру, Zip-дисковод интерфейса USB будет адресоваться так: `/dev/da0s4`. Некоторые съемные носители тоже поддерживают разбивку на разделы и обычно имеют один первичный раздел с номером 1 (`/dev/da0s1` или схожее имя) Другие носители не допускают разбивку на разделы и потому имеют идентификатор, обозначающий устройство в целом например `/dev/da0`

Файловые системы

Недостаточно знать коды разделов FreeBSD, чтобы получить доступ к файлам этих разделов. Любой раздел содержит еще и *файловую систему*. Это огромная иерархическая структура, которая "покрывает" раздел и позволяет операционной системе хранить файлы в нем В разных операционных системах применяются файловые системы различных типов Давайте познакомимся с ними поближе.

Два значения термина "файловая система"

Прежде всего следует отметить, что термин "файловая система" имеет два значения. Во-первых, под этим понимается совокупность структур данных, заполняющих раздел диска (иногда весь диск). Такие файловые системы имеют названия и поддерживаются конкретными операционными системами Например, говорят "файловая система ISO-9660" или "смонтировать файловую систему" (т е. сделать ее файлы доступными в некоем каталоге FreeBSD).

Другое значение термина "файловая система" — это логическая структура каталогов компьютера Например, можно сказать "при рекурсивном поиске файлы ищутся в любом каталоге файловой системы" В этом смысле файловая система скорее всего, занимает несколько разделов.

Как правило, смысл термина ясен из контекста Если может возникнуть неоднозначность, употребляют другие термины (*раздел* или *дерево каталогов*) либо делают уточнение (например "файловая система FAT")

Ключевые особенности файловой системы FreeBSD

Во FreeBSD используется файловая система *FFS (Fast File System)*. Ее еще называют *UFS (UNIX File System)*. В частности, в команде `mount` файловая система FFS помечается кодом `ufs`

По меркам компьютерной индустрии FFS — удивительно старая файловая система. Она берет начало в первых версиях BSD, выпущенных в Беркли. Как следует из названия ("быстрая файловая система"), разработчики FFS делали акцент на повышении скорости доступа к файлам. Для достижения этой цели применяется множество методик, включая эффективное расположение структур данных на диске, позволяющее минимизировать перемещения головки диска. Некоторые из этих методик не работают для современных жестких дисков EIDE и SCSI, однако производительность FFS остается на хорошем уровне.

UNIX-системы в целом, и FreeBSD в частности, реализованы в соответствии с предположением, что файловая система обладает определенными свойствами. Все они присущи и FFS.

- **Владельцы файлов**. Будучи многопользовательской операционной системой, FreeBSD следит за тем, кто создает файлы, т.е. кто является владельцем того или иного файла. Для этого каждому файлу назначается идентификатор пользователя (UID). Кроме того, файлы закрепляются за определенными группами пользователей посредством идентификаторов групп (GID).
- **Права доступа**. FreeBSD поддерживает трехуровневую систему прав доступа, определяющую права владельца, группы и остальных пользователей. Для каждой из этих категорий пользователей устанавливается возможность чтения, записи и выполнения (право выполнения помечает файл как программу или сценарий).
- **Специальные флаги**. Некоторые типы файлов требуют особой обработки. Например, каталоги — это файлы, содержащие указатели на другие файлы. Имена устройств, используемые для доступа к дисковым разделам, тоже трактуются как имена файлов. Эти примеры можно продолжать. Соответствующие флаги хранятся вместе с файлами, наряду с информацией о владельцах и правах доступа.
- **Характеристики имен файлов**. FFS поддерживает длинные имена файлов (до 255 символов) и распознает в них строчные и прописные буквы, числа, знаки препинания и даже управляющие символы. Сама FreeBSD различает регистр символов в именах файлов, т.е. прописные и строчные буквы трактуются по-разному. Это означает, что файл `FILE.TXT` отличается от файла `file.txt`, и оба они могут находиться в одном каталоге.

Эти свойства важны, поскольку многие сторонние файловые системы не обладают ими. Другими словами, при доступе к таким файловым системам FreeBSD должна имитировать то или иное свойство, чтобы работа с файлами велась единообразно. Например, FAT не поддерживает понятие владельца файла, поэтому FreeBSD вынуждена самостоятельно назначать владельца файлам FAT.

Другие важные файловые системы

При обмене данными с другой операционной системой приходится иметь дело с файловой системой иного типа. Ниже кратко описываются различные файловые системы, для монтирования которых нужно использовать разные команды.

Разновидности файловой системы FreeBSD

Поскольку FFS была разработана очень давно, она нашла применение во многих разновидностях UNIX. Ее поддерживают ближайшие "родственники" FreeBSD — OpenBSD и NetBSD, а также Solaris, NeXTStep, Mac OS X и ряд других UNIX-систем. Но во многих операционных системах используются собственные файловые системы. В Linux, например, это ext2fs. В коммерческих версиях UNIX широко применяются журнальные файловые системы.

Доступ к FFS разделам других BSD-систем с открытым исходным кодом не представляет трудностей, так как фундаментальная структура их файловых систем одинакова. Это, впрочем, не относится к другим реализациям FFS, особенно в коммерческих операционных системах. Эти реализации несовместимы с версией FFS, применяемой во FreeBSD. Если нужно обмениваться файлами с такой операционной системой, придется использовать другую файловую систему например FAT, в качестве общего знаменателя

FAT

В названии файловой системы FAT (File Allocation Table — таблица размещения файлов) отражена структура данных используемая для выделения дискового пространства файлам FAT получила широкое распространение в DOS, а позднее стала единственной файловой системой, поддерживаемой в Windows 9x/Me. Даже сегодня многие системы на базе Windows 2000/XP используют FAT, хотя в них доступны и более современные файловые системы (например, NTFS).

В соответствии с первоначальной задумкой, имена файлов FAT имеют ограниченную длину — 8 символов плюс необязательное трехсимвольное расширение (так называемое *соглашение 8.3*). Имена хранятся в виде прописных букв (допускаются также числа и знаки препинания), однако драйверы FAT во FreeBSD переводят их в нижний регистр. В Windows 95 компания Microsoft реализовала разновидность FAT, известную как VFAT, в которой поддерживаются длинные имена файлов смешанного регистра При этом сама VFAT не учитывает регистр, т.е. имена файлов, различающиеся только регистром символов, трактуются как идентичные. Важной особенностью VFAT является то, что она совместима со старыми системами С помощью VFAT можно сохранить файлы с длинными именами на диске FAT, а затем прочесть эти файлы в DOS где поддерживаются имена формата 8.3. В DOS видны только усеченные имена, но содержимое файлов остается неизменным

Записи FAT-таблиц бывают разной длины (это не связано с длиной имен файлов) Поддерживаются 12-, 16 и 32-разрядные записи, отсюда названия *FAT-12*, *FAT-16* и *FAT-32*. Больший размер записи соответствует диску большей емкости. Сегодня FAT-12 используется почти исключительно на дискетах FAT-16 применяется в небольших дисковых разделах (до 2 Гбайт, хотя после превышения порога 512 Мбайт лучше переключиться на FAT-32) Разделы, размер которых превышает 2 Гбайт должны содержать FAT-32, если только не предполагается работать с ними исключительно в среде Windows NT/2000/XP эти операционные системы могут обрабатывать разделы FAT-16 размером до 4 Гбайт.

Драйверы FAT во FreeBSD поддерживают как FAT, так и VFAT, причем все варианты FAT 12-, 16- и 32-разрядные Драйверы автоматически определяют размерность таблиц FAT, избавляя администратора от необходимости делать соответствующие установки. По умолчанию FreeBSD пытается самостоятельно выявить наличие VFAT и соответствующим образом смонтировать раздел, но можно задать и явные установки.

FAT не поддерживает концепции владельцев, прав доступа и специальных флагов в стиле UNIX, поэтому FreeBSD приходится их имитировать. Параметры имитации задаются с помощью опций монтирования

На заметку

Помимо традиционных средств монтирования, описываемых в следующем разделе, FreeBSD содержит также пакет `mtools`, предназначенный для организации доступа к FAT (особенно к дискетам) Пакет доступен в разделе `Emulators` программы `sysinstall` Команды имена которых начинаются на `m`, например `mdir` и `mcopy`, позволяют получить доступ к дискетам FAT

FAT поддерживается операционными системами компании Microsoft, всеми ос тальными ведущими операционными системами персональных компьютеров, а также большинством специализированных систем Вот почему дискеты съемные диски большой емкости и дисковые разделы предназначенные для обмена данными, чаще всего содержат FAT

NTFS

Компания Microsoft реализовала *NTFS (New Technology File System)* как часть Windows NT NTFS поддерживает длинные имена файлов (без учета регистра симво лов), концепцию владения файлами и списки контроля доступа Поскольку NTFS — предпочтительная файловая система для Windows NT, 2000 а теперь и XP, ее значи мость в ближайшем будущем должна возрасти.

Механизмы владения файлами и контроля доступа реализованы в NTFS не так, как во FreeBSD, поэтому они практически не поддерживаются Как и в случае FAT, можно заставить FreeBSD имитировать их при монтировании разделов NTFS, но только на уровне всей файловой системы в целом Поменять владельцев и права дос тупа отдельных файлов нельзя.

Формально FreeBSD допускает чтение и запись файлов NTFS, однако параметры записи являются столь строгими, что по сути NTFS используется как файловая систе ма, доступная только для чтения. Это еще больше снижает ее значимость во FreeBSD Если планируется обмениваться файлами с Windows NT/2000/XP на том же компью тере, лучше использовать разделы FAT. В крайнем случае можно смонтировать раздел NTFS и прочитать файлы из него

Ext2fs

Стандартная файловая система Linux называется *Second Extended Filesystem (ext2fs).* Она обладает теми же свойствами, что и FFS сильно напоминая ее с точки зрения считывания и записи файлов

К сожалению, средства поддержки ext2fs во FreeBSD не столь надежны, как следо вало бы ожидать. При чтении раздела ext2fs FreeBSD может перейти в нестабильное состояние и даже повредить файловую систему. Поддержка FFS в Linux тоже несо вершенна, так что если на компьютере установлены FreeBSD и Linux и нужно орга низовать обмен файлами между ними, придется создать раздел FAT

В настоящее время многие пользователи Linux переходят на другие файловые сис темы Одна из них, *Third Extended Filesystem (ext3fs)* — это журнальное расширение ext2fs Она относится к новому классу *журнальных файловых систем.* Ее разделы долж ны поддерживаться во FreeBSD, но пока что не рекомендуется монтировать разделы ext3fs в режиме чтения/записи, поскольку это может нарушить стабильность работы системы

Журнальные файловые системы

Журнальная файловая система содержит специальную структуру данных (*журнал*) предназначенную для фиксации изменений перед занесением их на диск. Идея заклю чается в том, что в случае сбоя системы или внезапного выключения питания наличие журнала изменений упрощает процедуру восстановления диска В некоторых файловых системах, например ext2fs, процесс восстановления протекает очень долго, поэтому благодаря журналу существенно сокращается время загрузки системы после сбоя.

В настоящее время используется много журнальных файловых систем Одна из них NTFS Стоит упомянуть также ext3fs (журнальная разновидность ext2fs), ReiserFS (новая файловая система, предназначенная для Linux), XFS (разработана компанией SGI для IRIX и уже перенесена в Linux) JFS (разработана компанией IBM

для AIX и OS/2 и тоже перенесена в Linux) и BeFS (разработана для BeOS). Ходят слухи о переносе некоторых из них во FreeBSD, но результата пока нет (информацию о планируемой реализации JFS во FreeBSD можно найти по адресу `http://sourceforge.net/projects/jfs4bsd`) Технические трудности, конечно, довольно велики, поэтому уйдет немало времени, прежде чем какая-нибудь из этих файловых систем начнет поддерживаться во FreeBSD Возможно одна из них со временем заменит FFS

На заметку

В FFS процесс восстановления протекает быстрее, чем в ext2fs, поэтому потребность в механизме журнализации не столь велика, как в Linux. В последних версиях FFS поддерживаются также *нежесткие обновления* Это другой способ обеспечения согласованности данных на случай системного сбоя, что еще больше снижает необходимость в журнальной файловой системе для FreeBSD

Файловые системы компакт-дисков

Компакт-диски представляют собой традиционное средство переноса не очень больших объемов данных Стандартный компакт-диск вмещает до 650 Мбайт данных и существуют способы повысить этот показатель до 700 Мбайт. Изначально компакт диски задумывались как носители, доступные только для чтения. Эта идея отражена в исходных файловых системах компакт-дисков Наиболее популярная из них, ISO-9660, существует в нескольких вариантах, часть которых налагает еще более строгие ограничения на имена файлов, чем DOS используется соглашение 8 3, но перечень допустимых знаков препинания меньше

UNIX-системы сияьно зависят от уникальных свойств своих файловых систем, по этому многие компакт-диски, созданные для UNIX, в том числе для FreeBSD, поддерживают расширение ISO-9660, известное как Rock Ridge Файлы записанные на компакт-дисках формата Rock Ridge, имеют владельцев и права доступа в стиле UNIX, длинные имена файлов и т.д Если FreeBSD обнаруживает что на компакт диске используются расширения Rock Ridge, она автоматически активизирует эти расширения при монтировании компакт-диска.

Для Windows 95 компания Microsoft разработала новую файловую систему извест ную как *Joliet.* Она поддерживает длинные имена файлов, но не механизмы владения и прав доступа в стиле UNIX. FreeBSD распознает разделы Joliet, однако если на компакт-диске используются расширения Rock Ridge и Joliet, то активизируются только первые.

С появлением записываемых компакт-дисков (CD-R и CD-RW) и оптических носителей большой емкости (DVD), ISO-9660 и родственные ей файловые системы столкнулись с серьезными препятствиями С целью устранения этих препятствий был разработан стандарт *UDF (Universal Disk Format).* В настоящее время ведется работа по его реализации во FreeBSD Последнюю информацию о проекте можно найти по адресу `http://people.freebsd org/ scottl/udf/`

Доступ к внешним файловым системам из FreeBSD

Важно не только знать существующие файловые системы и различия между ними, но и команды FreeBSD для работы с ними. Все они описываются в этом разделе. Здесь же будет рассказано о практических аспектах доступа к внешним файловым системам из FreeBSD

Монтирование внешних файловых систем

Для монтирования файловых систем во FreeBSD применяется команда `mount`. Она подключает файловую систему к *точке монтирования*, т. е. к каталогу, находящемуся где-то в иерархии каталогов FreeBSD. В общем виде синтаксис команды таков:

```
mount [опции, файл_устройства точка_монтирования
```

Аргумент *файл_устройства* — это имя устройства, на котором находится файловая система, например `/dev/ad0s4d` или `/dev/fd0`. Аргумент *точка_монтирования* — это каталог, через который будут доступны файлы. Работу команды `mount` можно модифицировать с помощью многочисленных опций. Например, опция `-a` заставляет команду смонтировать все файловые системы, перечисленные в файле `/etc/fstab`, а опция `-r` задает режим монтирования "только для чтения". Все эти опции описаны в главе 7. Здесь же мы остановимся на двух опциях, заслуживающих особого внимания.

- `-t` *тип*. Эта опция сообщает команде `mount`, какая именно файловая система содержится в монтируемом разделе или дисковом устройстве. Для каждого распознаваемого типа вызывается отдельная утилита с названием вида `mount_`*тип*, где *тип* - это аргумент опции `-t`. Например, в случае команды `mount -t msdosfs /dev/ad0s3 /windows` вызывается утилита `mount_msdosfs`, которая монтирует FAT-раздел `/dev/ad0s3` в каталоге `/windows` (до появления FreeBSD 5.0 тип `msdosfs` назывался `msdos`).
- `-o` *параметры*. С помощью этой опции можно задать различные параметры монтирования. Одни из них применяются ко всем файловым системам, другие же — к одной конкретной системе. Во втором случае можно задавать списки параметров и использовать оператор присваивания (`=`).

Если нужно смонтировать внешнюю файловую систему, воспользуйтесь командой `mount` с флагом `-t` или специализированной командой, предназначенной именно для этой файловой системы. Наиболее распространенные разновидности команды `mount`, соответствующие им файловые системы и параметры монтирования перечислены в табл. 4.4.

Таблица 4.4. Команды монтирования внешних файловых систем

Команда	*Файловая система*	*Распространенные параметры*
`mount_msdosfs` или `mount_msdos`	FAT (включая VFAT)	Параметр `-u` *UID* задает идентификатор пользователя для всех файлов, а параметр `-g` *GID* — аналогичный идентификатор группы. Параметр `-m` *маска* задает режим доступа для всех файлов. Флаг `-s` заставляет FreeBSD игнорировать и не генерировать длинные имена файлов VFAT. Флаг `-l`, наоборот, включает поддержку длинных имен (если ни один из двух флагов не указан, FreeBSD ищет длинные имена и, если они есть, использует их)
`mount_ntfs`	NTFS	Поддерживаются те же параметры `-u` *UID*, `-g` *GID* и `-m` *маска*, что и в предыдущем случае
`mount_ext2fs`	ext2fs	Специальные параметры отсутствуют
`mount_cd9660`	ISO-9660 (включая Rock Ridge), Joliet	Флаг `-j` заставляет FreeBSD игнорировать файловую систему Joliet, если она присутствует. Флаг `-r` заставляет FreeBSD игнорировать расширения Rock Ridge, если они используются
`mount_nfs`	NFS	Флаг `-2` включает использование NFS версии 2, а флаг `-3` — версии 3. Флаг `-s` заставляет FreeBSD возвращать коды ошибок после определенного числа неудачных попыток доступа

На заметку NFS (Network File System) — это сетевая файловая система широко используемая в сетях UNIX для организации совместного доступа к файлам. Подробнее она описана в главах 18 и 24

В качестве примера рассмотрим две команды:

```
# mount_msdosfs -u 1002 -s /dev/fd0 /mnt/floppy
# mount_cd9660  r /dev/acd0 /mnt/cdrom
```

Первая из них монтирует FAT дискету в катаяоге `/mnt/floppy` предоставляя дос туп ко всем файлам пользователю с идентификатором 1002 (каталог `/mnt/floppy` должен существовать) Опция `-s` означает игнорирование длинных имен файлов. Пользователи увидят только имена в формате 8.3; при копировании на дискету файла с длинным именем оно будет усечено до формата 8 3.

Вторая команда монтирует компакт-диск в каталоге `/mnt/cdrom` (он тоже должен существовать на момент ввода команды) Опция `-r` заставляет FreeBSD игнорировать расширения Rock Ridge Если на компакт-диске установлена файловая система Joliet, будут отображаться имена файлов соответствующего формата. В противном случае имена файлов будут соотвстствовать исходной спецификации ISO-9660 Обычно это означает формат 8.3, но некоторые разновидности ISO-9660 поддерживают имена файлов длиной до 32 символов одного регистра.

Эквивалентные команды `mount` выглядят так:

```
# mount -t msdosfs -o -u=1002,-s /dev/fd0 /mnt/floppy
# mount -t cd9660 -o -r /dev/acd0 /mnt/cdrom
```

Обратите внимание на то, что специализированным параметрам файловых систем предшествует опция `-o` В случае FAT-дискеты между `-u` и `1002` стоит не пробел, а знак равенства, а между `1002` и `s` — не пробел, а запятая. Такой формат позволяет в одной команде использовать как опции, относящиеся ко всем файловым системам (например, опция `r` задает доступ только для чтения), так и специализированные опции отдельных файловых систем (в случае файловой системы ISO-9660 та же опция `r` интерпретируется по-другому).

Если необходимо, чтобы какой-нибудь внешний раздел был доступен все время, создайте для него запись в файле `/etc/fstab`, как описано в главе 7 Там разрешается указывать как специализированные, так и общие параметры. FreeBSD автоматиче ски монтирует заданные файловые системы

Ограничения на испоьзоввние внешних файловых систем

Ранее уже упоминалось о некоторых особенностях, связанных с доступом к внеш ним файловым системам. Укажем наиболее общие моменты

- **Нестабильность системы** Драйверы некоторых файловых систем являются экспериментальными или плохо отлажены и могут вызвать сбой FreeBSD Сообщалось о подобных проблемах с ext2fs, например Использовать такие файловые системы следует с осторожностью Это не относится к файловым системам `msdosfs` и `cd9660`

- **Повреждение файловой системы** Даже если файловая система не вызовет сбоя FreeBSD может произоити повреждение файлов на диске, особенно если вклю чен режим чтения/записи Это относится к файловой системе ext2fs, средства поддержки которой во FreeBSD не очень хороши, но не к `msdosfs` Файловая система `cd9660` доступна только для чтения, так что проблем с ней не возникает. Если нужно создать файловую систему ISO-9660, воспользуйтесь специальной утилитой `mkisofs` совместно с еще одной утилитой, `cdrecord`, которая записывает файловую систему на чистый компакт-диск формата CD-R или CD-RW

- **Проблемы с именами файлов** Если монтируется файловая система FAT без расширений VFAT, то создавать файлы с длинными именами не разрешено Ни VFAT, ни NTFS не различают регистр символов, поэтому при копировании файлов, имена которых отличаются только регистром можно потерять файлы Можно также получить доступ к файлу по имени, регистр которого был бы не верным в случае FFS То же самое справедливо и в отношении Joliet, тогда как Rock Ridge больше напоминает FFS
- **Владельцы и права доступа** Администраторы-новички больше всего боятся проблем, связанных с правами доступа к файлам В случае однопользовательской рабочей станции можно задать в файле `/etc/fstab` параметр `-u=`*UID* для всех разделов FAT, гибких и съемных дисков, чтобы основной пользователь компьютера всегда имел доступ к файлам (подставьте вместо аргумента *UID* идентификатор этого пользователя). С помощью параметра `-m=777` можно назначить всем файлам режим доступа 777 (о режимах доступа рассказывается в главе 8), чтобы к ним мог обращаться любой пользователь. Если нужно разрешить доступ лишь определенной группе пользователей, задайте установку вида `-g=`*GID*, `-m=770` и убедитесь в том, что все пользователи, которые должны иметь доступ к файлам, входят в заданную группу (*GID*). Аналогичные опции применяются и в отношении разделов NTFS В разделах ext2fs владельцы обозначаются идентификаторами UID и GID и если в Linux и FreeBSD они не совпадают, будут наблюдаться странные эффекты Устранить эту проблему не так то просто, разве что согласовать между собой все используемые идентификаторы
- **Скорость доступа** Драйверы FFS во FreeBSD оптимизировались много лет, тогда как поддержка других файловых систем была реализована сравнительно недавно и гораздо менее оптимизирована. Следовательно, скорость доступа к внешним файловым системам может оказаться ниже, чем к "родным" разделам FFS

Средства поддержки внешних файловых систем оказываются чрезвычайно полезными во многих ситуациях. Особенно важна поддержка файловой системы ISO-9660 и связанных с ней расширений, а также поддержка FAT. Обе эти системы позволяют пользователям FreeBSD обмениваться файлами с другими системами

Доступ к файловым системам FreeBSD из других операционных систем

Иногда требуется получить доступ к файловым системам FreeBSD из другой операционной системы К примеру, вы загрузили Linux и внезапно обнаружили, что нужно прочесть файл, хранящийся в разделе FreeBSD. Бывает, что это единственный способ внести изменения в конфигурацию FreeBSD, если система не загружается (можно также воспользоваться аварийной дискетой, если она есть). Впрочем, файловая система FFS редко поддерживается "за пределами" UNIX поэтому возможности такого доступа ограничены

Поддержка со стороны других операционных систем

FFS используется во всех BSD-системах с открытым исходным кодом и ее реализации в этих системах совместимы. Таким образом, можно легко смонтировать разделы FreeBSD в OpenBSD или NetBSD. Достаточно ввести соответствующую команду `mount` или добавить описание раздела FreeBSD в файл `/etc/fstab` в OpenBSD или NetBSD

FFS поддерживается также во многих коммерческих разновидностях UNIX, включая Solaris. Но, как уже говорилось выше, файловая система менялась на протяжении многих лет, поэтому FFS-разделы FreeBSD не всегда удается смонтировать из за различий в реализации

Несмотря на то что Linux представляет собой повторную реализацию UNIX и поэтому не является прямым потомком BSD, она включает драйверы FFS, называемые драйверами *UFS* (соответствующая опция команды `mount` называется `-t ufs`) Эти драйверы поддерживают несколько различных вариантов FFS. Нужный вариант задается с помощью параметра `ufstype`, который в случае доступа к разделу FreeBSD должен быть равен `44bsd`. Следовательно, команда монтирования раздела FreeBSD будет выглядеть примерно так

```
# mount -t ufs -o ufstype=44bsd /dev/hda10 /mnt/freebsd
```

Важнее всего то что в Linux поддерживается BSD-схема разбивки на разделы Обычно пользователи работают с первичными расширенными и логическими разделами *x86* Перед обращением к BSD-разделам нужно вызвать меню настройки ядра и установить параметр Filesystems⇨Partition Types⇨BSD Disklabel (BSD Partition Tables) Support. Если по умолчанию он не был задан, придется перекомпилировать ядро Linux. После этого Linux начнет воспринимать первичный раздел FreeBSD как расширенный. Подразделы FreeBSD будут доступны под номерами 5 и выше или начиная с номера на единицу большего, чем номер последнего логического раздела

Если номер первичного раздела FreeBSD меньше, чем номер расширенного раздела, то добавление или удаление подразделов FreeBSD приведет к изменению нумерации логических разделов, поскольку Linux нумерует логические разделы и разделы FreeBSD в соответствии с номером расширенного или первичного раздела, в котором они находятся. Такое изменение, в свою очередь, может привести к тому, что система откажется загружаться до тех пор, пока изменения не будут отражены в файле `/etc/fstab`. Проблема не возникает, если номер расширенного раздела меньше, чем номер первичного раздела FreeBSD

Средства поддержки FFS в Linux далеки от совершенства В частности режим чтения/записи все еще считается экспериментальным и включается отдельно от режима чтения Это означает что разделы FreeBSD могут оказаться недоступными для записи, если ядро компилировалось с базовыми средствами поддержки FFS. Включать режим чтения/записи следует с осторожностью так как существует вероятность повреждения раздела

Особенности внешнего доступа

Те же ограничения о которых говорилось в контексте доступа к внешним файловым системам из FreeBSD, касаются и доступа к файловым системам FreeBSD из других операционных систем, пусть даже формально они поддерживают FFS Все таки за многие годы появилось множество плохо совместимых реализаций FFS. Даже если файловую систему удалось смонтировать, может возникнуть несогласованность в отношении значений UID и GID Чтобы свести эти проблемы к минимуму нужно заранее продумать схему идентификации пользователей и групп

Более опасна возможность повреждения файлов, особенно если обращаться к разделам FFS в режиме чтения/записи из Linux. Создайте резервные копии файлов, прежде чем осуществлять запись в них Режим чтения разделов FFS менее опасен по этому чтение файлов FreeBSD в Linux обычно не вызывает проблем

Запуск сторонних программ во FreeBSD

Для FreeBSD написано много программного обеспечения, но иногда приходится запускать программы из других операционных систем. Вместо того чтобы покупать второй компьютер или инсталлировать вторую ОС в режиме мультисистемной загрузки, можно воспользоваться специальными утилитами для запуска во FreeBSD сторонних программ. Надежнее всего работают средства эмуляции Linux. Существуют также средства запуска Windows программ и эмуляции виртуальных машин. Последние позволяют инсталлировать любую операционную систему "внутри" FreeBSD.

Запуск программ Linux

Несмотря на разное происхождение, FreeBSD и Linux довольно похожи. Во многих случаях один и тот же исходный код может компилироваться и выполняться в любой из систем. Но не все программы распространяются с открытым исходным кодом, а некоторые коммерческие программы доступны в Linux, но не во FreeBSD. По этой причине во FreeBSD предусмотрены средства запуска Linux программ. Сюда входит ряд библиотек и вспомогательных пакетов. После их инсталляции программы смогут работать во FreeBSD почти так же, как и в Linux. FreeBSD и Linux используют практически идентичные системные вызовы, поэтому эмулировать приходится не так уж много компонентов. Как следствие, производительность Linux-программ остается практически неизменной. Вместе с тем возрастают требования к памяти, поскольку Linux-программам нужны совместимые системные библиотеки, многие из которых дублируют функции стандартных библиотек FreeBSD.

Подготовка к запуску

В режиме базовой инсталляции FreeBSD не конфигурируется на запуск Linux программ. Чтобы включить такую поддержку, следуйте приведенным ниже инструкциям.

1. **Инсталлируйте базовую коллекцию переносимых приложений Linux.** Пакет `linux_base` содержит ключевые библиотеки Linux и вспомогательные утилиты. Большинство из них находится в каталоге `/compat/linux`. Порядок инсталляции пакетов описан в главе 11.
2. **Настройте режим распознавания доменных имен.** В стандартной конфигурации некоторые Linux-программы испытывают трудности с преобразованием доменных имен в IP-адреса. Если возникают проблемы, отредактируйте файл `/compat/linux/etc/host.conf`, чтобы он содержал следующие строки:

```
order hosts bind
multi on
```

3. **Включите режим совместимости с Linux.** Наберите `linux`, чтобы включить режим совместимости с Linux. После этого можно запускать Linux-программы. Если необходимо, чтобы такой режим включался автоматически при загрузке системы, добавьте следующую строку в файл `/etc/rc.conf`:

```
linux_enable="YES"
```

С этого момента FreeBSD должна поддерживать библиотеки Linux. Чтобы убедиться, воспользуйтесь командой `kldstat`, которая выдает информацию о загруженных модулях ядра, включая модуль совместимости с Linux. Если все сделано правильно, результат должен получиться примерно таким (строк будет больше, если в системе есть другие модули):

```
$ kldstat
Id Refs Address    Size     Name
 1    5 0xc0100000 3ef2dc   kernel
 2    1 0xc1cb2000 16000    linux.ko
```

Главное, чтобы присутствовала строка с модулем `linux ko`

Переконфигурирование и запуск

FreeBSD и Linux поддерживают один и тот же базовый формат исполняемых файлов: *ELF (Executable and Linking Format)* Но детали могут отличаться, поэтому иногда FreeBSD не распознает исполняемые файлы Linux. Если система сообщает о том, что формат исполняемого файла некорректен, воспользуйтесь утилитой `brandelf`, чтобы нужным образом пометить файл. Команда должна быть примерно такой

```
# brandelf -t Linux исполняемый_файл
```

Теперь программа будет работать так как если бы она была написана для FreeBSD Введите ее имя и посмотрите, что произойдет. Текстовая программа должна отобразить какие-то данные на экране, а X-программа откроет окно и начнет выполняться в нем. На основании пользовательского интерфейса нельзя определить для какой системы — Linux или FreeBSD — написана программа

К сожалению, Linux-программам часто нужны библиотеки, которые не инсталлированы во FreeBSD. Точнее, им нужны библиотеки *Linux*. Недостаточно предоставить FreeBSD-версию такой библиотеки Большую часть пакета `linux_base` составляют именно такие библиотеки. Но этот пакет нельзя считать полным, поэтому иногда приходится отслеживать, какие именно версии библиотек нужны

Для определения недостающих библиотек предназначена утилита `ldd` имеющаяся как в Linux, так и во FreeBSD. Ниже показан отчет, выдаваемый утилитой во FreeBSD 4 4 для команды `ls` из Caldera OpenLinux 3 1

```
$ ldd ls
ls:
ls: /lib/libc.so.6: version `GLIBC_2.2' not found (required by ls
        libc.so.6 => /lib/libc.so.6 (0x28068000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x28053000
```

На заметку В этом примере использована более новая версия Linux и более старая версия FreeBSD с целью продемонстрировать, что старые библиотеки (из набора средств совместимости с Linux во FreeBSD 4 4) могут не работать с новыми исполняемыми файлами

В данном случае команда `ls` зависит от двух библиотек. `libc.so 6` и `ld linux.so.2`. Обе они существуют, но версия `libc.so.6` слишком старая, отсюда сообщение о том, что библиотека `GLIBC 2.2` не найдена. Может также оказаться, что требуемой библиотеки вообще нет во FreeBSD.

Как бы там ни было, решение заключается в том, чтобы скопировать необходимые библиотеки из Linux (или из пакета Linux-библиотек) в каталог `/compat/linux` FreeBSD В нашем случае файл `/lib/libc.so.6` в Caldera Linux является символической ссылкой на файл `/lib/libc-2.2.1.so` Чтобы все работало правильно, выполните следующее.

1. Скопируйте реальный библиотечный файл (`/lib/libc-2 2 1.so`) в каталог `/compat/linux` Другими словами файл `libc-2.2.1.so` должен быть помещен в каталог `/compat/linux/lib`
2. Создайте символическую ссылку, аналогичную той, которая существует в Linux. Если необходимо, замените существующую ссылку связанную со старой библиотекой.

В случае необходимости проделайте то же самое для остальных библиотек Иногда, разобравшись с одной проблемой, обнаруживаешь следующую Например, в случае команды `ls` из Caldera 3 1 выясняется, что после обновления библиотеки `libc so.6` нужно обновить и библиотеку `ld linux so.2` Когда все будет завершено, программа должна заработать Некоторые программы выдают сообщения о некритических ошибках В нашем примере команда `ls` пожалуется, что системные вызовы `fstat64` и `setup` являются устаревшими

Вместо того чтобы обновлять библиотеки от случая к случаю, поищите обновленную версию пакета `linux base` Его гораздо проще инсталлировать

Режим бинарной совместимости с Linux имеет и другие ограничения, которых следует остерегаться Не все средства Linux доступны во FreeBSD, а некоторые реализованы по-другому. Важным моментом является поддержка файловой системы `/proc` Она предоставляет программам доступ к системной информации, которая в Linux и FreeBSD различна Если Linux-программа собирает информацию из файловой системы `/proc`, то она может работать некорректно во FreeBSD

В целом режим бинарной совместимости с Linux очень удобен для запуска коммерческих приложений, таких как WordPerfect, StarOffice, Oracle, VMware и Quake, т.е. программ с патентованым кодом Если же есть доступ к исходному коду, то бинарная совместимость не важна

Использование пакета WINE для запуска приложений Windows

Большинство популярных программ, которые не работают во FreeBSD, не работают и в Linux, и в других разновидностях UNIX В основном это приложения Windows. Операционная система Windows заимствовала некоторые концепции и даже часть кода из UNIX, но при этом реализована совершенно иначе, поэтому во FreeBSD не так-то просто запустить программу, написанную для Windows. Ведутся проекты по внедрению во FreeBSD средств поддержки Windows. Наиболее важный из этих проектов — *WINE* Его название расшифровывается как "Windows Emulator" (эмулятор Windows) или "WINE Is Not an Emulator" ("WINE - не эмулятор"), в зависимости от точки зрения WINE представляет собой систему с открытым исходным кодом, реализующую функции Windows API в UNIX-системах, включая FreeBSD. Это означает, что программы могут использовать системные вызовы Windows для открытия диалоговых окон, отображения текста и т.д. Поддерживается также возможность запуска исполняемых файлов Windows, для которых формируется соответствующая среда

Разработчики WINE не лишены амбиций, если учесть, с кем они пытаются тягаться. Ведь компания Microsoft постоянно модифицирует и расширяет набор функций Windows API, а часть из них вообще не документирована Поэтому проект WINE еще далек от совершенства, несмотря на почти десятилетнюю историю Он, конечно, удобен для запуска ряда программ, среди которых есть довольно важные приложения, но охватывает далеко не все программы, написанные для Windows. Последняя информация о проекте, включая список протестированных программ доступна на Web-узле `www.winehq.com`

Инсталляция и конфигурирование пакета WINE

Для инсталляции заранее скомпилированного пакета проще всего воспользоваться программой `sysinstall` WINE указан в категории `Emulators`. Можно также загрузить исходный код пакета с Web-узла и скомпилировать его самостоятельно Архив исходных кодов обновляется каждую неделю-две, тогда как пакет, включенный в состав FreeBSD, может быть полугодичной давности

Пакет нужно не только инсталлировать, но и должным образом сконфигурировать Конфигурационный файл называется `wine.conf`. По умолчанию он записывается в каталог `/usr/local/etc`. В дистрибутив пакета входит также файл `wine.conf.sample`, который можно просто скопировать в нужный каталог и переименовать в `wine.conf`. Но сама программа WINE не всегда заглядывает в этот каталог. Сначала проверяется файл `config` в подкаталоге `$HOME/.wine`. Скопируйте файл `/usr/local/etc/wine.conf` под именем `/.wine/config` или создайте соответствующую символическую ссылку Лучше работать с копиями файлов, так как это позволит пользователям по разному конфигурировать программу

По формату конфигурационный файл WINE напоминает старый файл `WIN.TNI` из Windows 3 1 Его содержимое разбито на разделы, названия которых указаны в квадратных скобках В этих разделах определяются связи между каталогами FreeBSD и логическими дисками Windows, указываются доступные последовательные порты, аппаратные устройства и т д В стандартный конфигурационный файл почти наверняка придется вносить изменения Наиболее важные из них описаны ниже

- **Диск C:**. Windows загружается, по крайней мере частично, из раздела, который идентифицируется как диск `C:` В нем находятся специальные каталоги, такие как `WINDOWS` и `WINDOWS\SYSTEM` Укажите в разделе `[Drive C]` конфигурационного файла WINE, где именно смонтирован раздел Windows. Если, к примеру, это каталог `/windows`, задайте его с помощью параметра `Path` (в большинстве случаев по умолчанию принят каталог `/c`) Если режим мультисистемной загрузки не используется, все равно укажите путь к диску `C:` и создайте соответствующие каталоги, даже если они пусты Можно, например, ввести **`mkdir -p /windows/windows/system`**. Когда WINE не находит нужные каталоги ия одна Windows-программа не запускается
- **Диск A:**. Windows обозначает дисковод гибких дисков как `A:`. По умолчанию в конфигурационном файле WINE диск `A:` связан с каталогом `/mnt/fd0`, который не существует во FreeBSD Укажите вместо него точку монтирования чаще всего используемую для подключения дисковода гибких дисков
- **Диски данных** Нужно указать программе WINE, в каких каталогах содержатся программные файлы Если они находятся в начальном каталоге пользователя, то стандартные установки раздела `[Drive F]` подойдут В противном случае придется создать новое определение диска с уникальной буквенной меткой. Можно скопировать содержимое раздела `[Drive C]` и модифицировать его по своему усмотрению Не забудьте поменять буквенную метку а также спецификации `Path` и `Label`.
- **Параметры DLL** Библиотеки DLL (Dynamic Link Library) напоминают библиотеки FreeBSD. Они тоже содержат код, используемый множеством программ. В Windows есть большое количество системных библиотек DLL а некоторые программы поставляются с собственными библиотеками. WINE может работать с этими библиотеками, но в некоторых случаях предпочтительнее использовать специальные реализации DLL-функций. В разделе `[DLLOverrides]` сообщается, какие версии библиотек DLL — встроенные (WINE) или исходные (Windows) — следует применять Во втором случае библиотеки необходимо сделать доступными (обычно в каталоге `C:\WINDOWS\SYSTEM`).
- **Параметры X среды** В разделе `[x11drv]` заданы различные параметры касающиеся взаимодействия программы WINE с X-сервером. В частности, параметр `Managed` определяет, должна ли программа WINE создавать собственные рамки окна (`N`) или нужно использовать стандартный диспетчер окон (`Y`). Параметр `Desktop` заставляет программу создать окно для запуска вызываемого приложения

В конфигурационном файле содержится множество других параметров, но их рас смотрение выходит за рамки этой главы Подробности можно узнать в документации или в комментариях к стандартному конфигурационному файлу.

Запускаемые приложения Windows должны быть доступны программе WINE Обычно для этого создаются определения дисков данных как описывалось выше Можно скопировать приложения Windows во FreeBSD или запускать их с файл сервера, при условии, что его каталоги разрешено монтировать

Во FreeBSD можно запустить серверы Samba и NFS, как описано в главе 18, чтобы сделать программы доступными как в Windows, так и во FreeBSD Естественно, нужно предварительно убедиться в том что лицензионные требования разрешают это

Запуск приложений Windows

Чтобы вызвать оболочку WINE, нужно ввести ее имя в командной строке (`wine`) а затем указать имя вызываемой программы К примеру, следующая команда запускает игру Solitaire (разновидность пасьянса)

```
$ wine /windows/windows/sol.exe
```

При этом подразумевается, что диск `C:` смонтирован в каталоге `/windows`

При первом запуске программы WINE процесс инициализации довольно продол жителен (больше минуты), поскольку загружаются нужные шрифты и создаются конфигурационные файлы в каталоге `~/.wine` Впоследствии программа запускается гораздо быстрее На рис 4 2 изображен результат выполнения предыдущей команды (окно создается стандартным диспетчером окон X Window System)

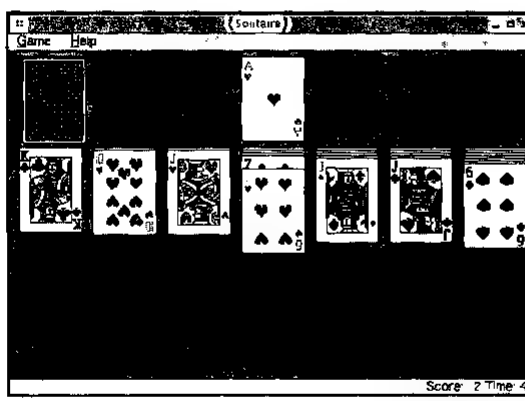


Рис 4 2 Программа WINE позволяет запускать приложения Windows в среде FreeBSD

Некоторые приложения Windows хорошо работают при стандартной конфигурации WINE Другие вообще не запускаются. Промежуточные случаи самые неприятные: программа может запуститься, но лишь при наличии нестандартных параметров конфигурации На Web-узле WINE приведена информация о таких программах Можно

также обратиться за помощью в телеконференцию `comp emulators.ms-windows wine` Как правило, чем меньше, древнее и универсальнее программа, тем легче ее запускать с помощью WINE

Использование эмулятора

Как уже упоминалось выше, акроним WINE имеет два толкования "Windows Emulator" и "WINE Is Not an Emulator". Такое разногласие возникло из-за того, что термин *эмулятор* понимается по-разному. Для одних это программа, которая позволяет запускать приложения, написанные для другой операционной системы. Для других это программа, реализующая низкоуровневые аппаратные функции. По мнению большинства пользователей, режим совместимости с Linux во FreeBSD не считается эмуляцией, а программа WINE является эмулятором лишь частично Но как бы там ни было, во FreeBSD поддерживаются пакеты, совершенно очевидно являющиеся эмуляторами. Многие из них перечислены в списке `Emulators` программы `sysinstall`

Во FreeBSD эмуляторы выполняют различные специализированные функции Наиболее простые из них имитируют игровые консоли и среду старых компьютеров, таких как Commodore 64 Другие эмуляторы позволяют запустить целую операционную систему в рамках "виртуального" компьютера работающего под управлением FreeBSD К примеру, программа VMware (`www.vmware.com`) эмулирует среду компьютера x86, в которой можно запустить DOS, большинство версий Windows, Linux, FreeBSD и другие операционные системы Программа прекрасно справляется со своей задачей, но во многом это достигается за счет снижения производительности В эмулируемой среде операционная система работает медленно, особенно в графическом режиме Тем не менее программа VMware обеспечивает гораздо лучшую совместимость, чем WINE.

Вообще говоря, программа VMware предназначена не для FreeBSD, а для Linux Во FreeBSD ее придется запускать в режиме совместимости Пакет VMware, инсталлируемый программой `sysinstall`, включает этот режим, но при этом устанавливает демонстрационную версию программы По окончании демонстрационного периода будет предложено приобрести лицензию у компании VMware

Резюме

FreeBSD-системы часто сосуществуют с другими операционными системами Простейший случай — чтение дискет DOS или компакт-дисков ISO-9660. В других случаях FreeBSD инсталлируется вместе с другой системой на одном компьютере, и приходится конфигурировать режим мультисистемной загрузки Необходим также надежный канал обмена файлами между двумя системами Иногда во FreeBSD приходится запускать программы, написанные для других операционных систем. К счастью, во FreeBSD есть средства, реализующие такую возможность. В частности, поддерживается файловая система FAT, используемая в DOS и Windows. Это дает возможность обмениваться данными, хранящимися на дискетах и жестких дисках. Имеются средства бинарной совместимости с Linux, позволяющие запускать во FreeBSD коммерческие приложения, написанные для Linux Программа WINE обеспечивает поддержку Windows-приложений, хотя и далеко не всех Есть и другие эмуляторы, с помощью которых можно запускать программы, недоступные во FreeBSD

Полный справочник по

Часть II

Основы системного администрирования

Полный справочник по

Глава 6

Средства системного администрирования

Вторая часть этой книги посвящена *системному администрированию*, т.е. действиям, которые связаны с изменением операционной системы в целом, инсталлированных в ней программ или учетных записей пользователей. Административные задачи рассматриваются также в частях III, IV и VI, здесь же речь пойдет об основах наиболее фундаментальных задачах

Данная глава представляет собой введение в системное администрирование. В ней описаны базовые средства и файлы, используемые для решения административных задач. В первую очередь читатели познакомятся с содержимым каталога `/etc`, в котором хранятся основные административные файлы. Поскольку задачи администратора зачастую сводятся к редактированию этих файлов, необходимо уметь работать хотя бы с одним из текстовых редакторов FreeBSD. В главе рассматриваются наиболее популярные из них (`vi` и gEdit) и приводятся ссылки на ряд других. Далее описываются основные административные команды, которые выдают важную информацию о системе или позволяют менять общесистемные параметры. В конце главы вкратце рассказывается о графических административных утилитах. В среде UNIX к ним традиционно относятся с пренебрежением, но в определенных обстоятельствах они весьма полезны, например для администраторов-новичков

Каталог /etc и его содержимое

При размещении файлов программ, написанных для FreeBSD, должны соблюдаться определенные правила (подробнее об этом рассказывается в разделе "FHS" главы 7). Одно из таких правил гласит, что системные конфигурационные файлы должны находиться в каталоге `/etc`. В общем случае имя файла совпадает с названием программы, которой он управляет, или же программа создает подкаталог с таким же именем, в котором хранится несколько конфигурационных файлов. В этом разделе мы поговорим о наиболее важных конфигурационных файлах, разбив их на три класса: 1) сценарии запуска системы, контролирующие инициализацию FreeBSD после загрузки ядра; 2) серверные конфигурационные файлы, управляющие функционированием различных серверов; 3) вспомогательные конфигурационные файлы, управляющие пользовательскими приложениями или содержащие общесистемные установки

На заметку

Не все программы хранят конфигурационные файлы в каталоге `/etc`. У одних вообще нет конфигурационных файлов. Другие записывают их в нетрадиционные каталоги. Третьи используют каталог `/usr/local/etc`, а не `/etc`. Четвертым достаточно конфигурационных файлов, находящихся в начальных каталогах пользователей. Чаще всего имена таких файлов начинаются с точки (это позволяет не включать их в обычный листинг команды `ls`)

Сценарии запуска системы

В главе 3 описывалась процедура запуска FreeBSD. Давайте вспомним еще раз, как все происходит. В первую очередь выполняется код, записанный в BIOS. Затем вызывается системный загрузчик, его задача — загрузить ядро FreeBSD. Ядро обращается к программе `/sbin/init`, которая контролирует последующий запуск системы.

Во FreeBSD, как и в других операционных системах, любая программа запускается другой программой. Запускающая программа называется *предком*, запускаемая — *потомком*. Выполняемая программа называется *процессом* и у каждого процесса есть

свой уникальный идентификатор (process ID PID) О процессе говорят, что он *порождается*

Это терминологическое отступление было сделано потому что программа `init` является предком (прямым или косвенным) всех остальных процессов. Ее конфигурационный файл называется `/etc/rc` (вообще говоря, это сценарий интерпретатора команд `/bin/sh`) Он определяет, какие процессы порождаются при запуске системы С его помощью можно контролировать поведение системы

На заметку

Во FreeBSD применяется модель запуска общая для всех BSD систем В других UNIX-подобных системах, в частности дистрибутивах Linux, используется совершенно иная модель, основанная на сценариях System V. В BSD-схеме всего несколько крупных сценариев, тогда как в System V множество более мелких сценариев. В System V поддерживается также понятие *уровней выполнения* Каждый такой уровень образуется уникальной совокупностью выполняемых серверов и других процессов В модели BSD и следовательно, во FreeBSD уровней выполнения нет.

Сценарий `/etc/rc` выполняет множество функций, среди которых проверка дисков, если они не были корректно демонтированы, удаление старых файлов из каталога `/var` и активизация области подкачки Целый ряд функций делегируется другим сценариям каталога `/etc`, имена которых начинаются с префикса `rc`, например `rc.serial` и `rc network` Все они контролируют процедуру запуска FreeBSD. Это и есть основные конфигурационные файлы системы.

Чтобы вносить изменения в системные сценарии, нужно знать принципы написания сценариев для интерпретатора команд Об этом будет рассказываться в главе 31 Впрочем, иногда изменения вполне тривиальны и не требуют специальных знаний На пример, бывает достаточно поменять название программы вызываемой из сценария.

Внимание

Модификация сценариев запуска FreeBSD — потенциально опасное занятие. Если допустить ошибку, компьютер может не загрузиться Это особенно опасно в случае важных серверов, которые работают без перерыва дни, недели и даже месяцы По прошествии столь долгого времени сложно вспомнить, какое именно изменение оказалось фатальным Перед редактированием любого системного сценария рекомендуется создать его резервную копию, а сразу после редактирования — перезагрузить систему. Если возникнет ошибка, загрузитесь с аварийной дискеты, восстановите исходный фаил и сделайте новую попытку

С запуском системы связаны также сценарии каталога `/usr/local/etc/rc.d` и сценарий `/etc/rc local` Они служат одинаковым целям, но первые являются предпочтительным средством в последних версиях FreeBSD. Подробнее о настройке этих сценариев рассказывается в главе 6

На заметку

Слово "local" в названии файла `/etc/rc local` и каталога `/usr/local` указывает на системно-зависимую природу соответствующих сценариев в противоположность стандартным сценариям запуска FreeBSD.

Серверные конфигурационные файлы

Второй класс файлов, находящихся в каталоге `/etc` — это файлы конфигурирования серверов. Многие из них подробнее описываются в последующих главах в ос

новном - в части IV Ниже кратко указано назначение наиболее важных среди сер верных конфигурационных файлов и каталогов

- **Файлы X-сервера.** Файл `/etc/XF86Config` и файлы каталога `/etc/X11` управляют X-сервером пакета XFree86 и связанными с ним программами, такими как X Display Manager (XDM) Подробнее об этих файлах рассказывается в главе 13
- **Файлы Apache** В каталоге `/usr/local/etc/apache` содержатся конфигурационные файлы популярного Web сервера Apache. Основной из них — `httpd.conf` Конфигурирование сервера Apache рассматривается в главе 20.
- **`/etc/exports`** Этот файл определяет каталоги доступные через файловый сервер NFS (см главу 18)
- **Файлы TCP Wrappers.** Программа TCP Wrappers — важное средство защиты используемое многими серверами Она конфигурируется посредством файлов `/etc/hosts.allow` и `/etc/hosts.deny` (см главу 29).
- **Файлы надежных узлов.** Файл `/etc/hosts.equiv` используется несколькими серверами для определения того, какие удаленные компьютеры имеют доступ к локальным серверам. Файл `/etc/hosts lpd` служит аналогичным целям но только в отношении стандартного сервера печати FreeBSD
- **`/etc/inetd.conf`** Этот файл, контролирующий работу стандартного *супердемона* `inetd`, чрезвычайно важен Демон замещает собой другие серверы, запуская их только в случае необходимости Такая конфигурация снижает потребление ресурсов памяти, если на компьютере установлено небольшое количество редко используемых серверов. Демон `inetd` поддерживает программу TCP Wrappers, с помощью которой можно реализовать унифицированный контроль доступа ко всем серверам Подробнее файл `inetd.conf` рассматривается в главе 16
- **`/etc/mail`** В этом каталоге хранятся конфигурационные и управляющие файлы программы `sendmail` — стандартного почтового SMTP-сервера FreeBSD Основным является конфигурационный файл `/etc/mail/sendmail.cf`, но он, в свою очередь генерируется на основании файла `/etc/mail/freebsd.mc` (см главу 19)
- **`/usr/local/etc/smb conf`** Это конфигурационный файл программы Samba реализующей функции файлового сервера и сервера печати для клиентов Microsoft Windows По умолчанию в процессе инсталляции программы создается файл `/usr/local/etc/smb.conf.default`, который нужно скопировать под именем `/usr/local/etc/smb.conf` и модифицировать так, как описано в главе 18.
- **Файлы SSH** Пакет SSH позволяет шифровать каналы удаленного доступа В каталоге `/etc/ssh` содержатся файлы конфигурации как клиентов (`ssh config`), так и серверов (`sshd_config`) SSH

Это далеко не полный список — едва ли можно описать все существующие серверы. Не все серверы помещают свои конфигурационные файлы в каталог `/etc` или `/usr/local/etc` Подробности всегда можно узнать в документации к серверу. Кроме того, некоторые из перечисленных выше файлов могут отсутствовать в системе, если соответствующий сервер не был инсталлирован.

Для большинства серверов требуется не только конфигурирование, но и правильная организация запуска Для этого нужно либо создать сценарий запуска сервера в каталоге `/usr/local/etc/rc.d`, либо добавить вызов сервера в файл `/etc/inetd conf`, как описано в главе 16

Вспомогательные конфигурационные файлы

Не все конфигурационные файлы связаны с конкретными серверами Некоторые из них задают общесистемные параметры или настройки для целого ряда серверов Таковыми, в частности, являются конфигурационные файлы программы TCP Wrappers. Ниже перечислены другие важные конфигурационные файлы

- **/etc/crontab** Во FreeBSD есть утилита `crontab`, которая используется для регулярного запуска программ в запланированное время Файлы заданий, называемые `crontab`-файлами, позволяют сконфигурировать систему на автоматическое выполнение повседневных регламентных операций Фаил `/etc/crontab` управляет системными заданиями Его формат описан в главе 6
- **/etc/dhclient conf** На компьютерах, являющихся клиентами DHCP, выполняется программа `dhclient`, которая автоматически загружает IP-адрес и другие сетевые параметры с DHCP-сервера (см. главу 14) Конфигурационный файл этой программы называется `/etc/dhclient.conf` Обычно его не приходится редактировать
- **/etc/fstab**. Это ключевой файл, связанный с управлением разделами и файловыми системами. В нем хранится таблица разделов с указанием типа файловой системы каждого раздела и точки монтирования раздела к дереву каталогов FreeBSD Подробнее об этом рассказывается в главе 7
- **/etc/printcap** Этот файл контролирует работу с принтерами В нем приведен список поддерживаемых принтеров, указано какие устройства используются для подключения к ним какие программы обрабатывают посылаемые на принтер данные и т.д. О конфигурировании принтера речь пойдет в главе 9
- **/etc/resolv.conf** Этот файл сообщает системе, какие серверы имен отвечают за преобразование доменных имен в IP-адреса (см. главу 14).
- **Файлы пользовательских и групповых учетных записей.** Файлы `/etc/passwd` `/etc/master.passwd` и `/etc/group` хранят информацию об учетных записях пользователей и групп. Эти файлы приходится редактировать при изменении учетных записей (лучше делать это не вручную с помощью текстового редактора, а использовать специализированные утилиты)

Текстовые редакторы

Прежде чем приступать к редактированию конфигурационных файлов, необходимо изучить средства редактирования. В этом разделе читатели узнают какие текстовые редакторы доступны во FreeBSD, и познакомятся с двумя из них

На заметку

Помимо текстовых редакторов существует класс программ, называемых *текстовыми процессорами*. Текстовые редакторы предназначены для работы с обычными текстовыми файлами, имеющими минимум форматирования и один стандартный шрифт. Текстовые процессоры поддерживают специальное форматирование и позволяют осуществлять сложную обработку текста. Впрочем, аналогичные возможности открываются и в текстовом редакторе, если использовать его совместно с пакетом средств форматирования, таким как TeX. Подробнее о средствах обработки текста, доступных во FreeBSD рассказывается в главе 25.

Текстовые редакторы можно классифицировать по-разному. Мы будем различать редакторы, работающие в текстовом и графическом режимах. Часто недооценивают важность редакторов первого типа, но именно они приходят на помощь в экстренных ситуациях, когда графическая оболочка оказывается недоступной

Редакторы текстового режима

Такие редакторы способны работать не только в чисто текстовой среде, но и в терминальном окне. Как правило, одного такого редактора вполне достаточно, но не помешает хотя бы бегло познакомиться с другими редакторами, так как они могут быть по умолчанию сконфигурированы на выполнение определенных операций или входить в состав аварийного комплекта программ, записываемых на загрузочную дискету

Ниже перечислены наиболее популярные редакторы.

- `vi` Это один из старейших текстовых редакторов UNIX У него довольно не привычный, трехрежимный, пользовательский интерфейс, который многие находят весьма неудобным Но несмотря на это редактор `vi` очень популярен За его внешней простотой скрываются широкие функциональные возможности поэтому у редактора немало поклонников Кроме того, небольшой размер редактора позволяет помещать его на загрузочные дискеты. Вообще говоря, `vi` — это целое семейство схожих редакторов, таких как Vim и Elvis Но здесь рассматривать такие подробности не представляется целесообразным
- **Emacs** Во многих отношениях редакторы Emacs и `vi` диаметрально противоположны. Сложность и громоздкость первого разительно контрастируют с простотой и небольшим размером второго. Существует даже шутка, что UNIX — это просто среда для запуска программ, а Emacs и есть операционная система Emacs может функционировать не только в качестве текстового редактора, но и как интегрированная среда разработки, программа чтения электронной почты Web-броузер и т д. Существует также его графический вариант XEmacs.
- **Усеченные версии редактора Emacs** В распоряжении пользователей, которым нравится редактор Emacs и которые хотели бы иметь его на загрузочных дискетах, есть усеченные "клоны" этого редактора `jed`, `joe` `jove`, `uemacs`. Они могут послужить альтернативой редактору `vi`
- **Easy Editor** Это небольшой редактор, который используется для редактирования файлов в процессе инсталляции FreeBSD (см. главу 2) Команда вызова редактора называется `ee`

Это далеко не полный перечень текстовых редакторов, которых за последние годы появилось очень много Просмотреть список доступных редакторов можно в разделе `Editors` меню программных пакетов программы `sysinstall`. Там перечислены редакторы не только текстового, но и графического режима, а также различные вспомогательные программы

В контексте текстовых редакторов стоит рассмотреть дополнительный класс программ, называемых *программами постраничной разбивки* Они выводят текстовые файлы на экран по одной странице за раз. Первой стандартной программой данного класса в UNIX была `more`; она и сейчас входит в состав FreeBSD. Именно она, кстати, применяется по умолчанию программой `man` для отображения страниц интерактивного руководства Но со временем недостатки программы `more` стали очевидными и появилась ее усовершенствованная версия названная `less`, т.е. "меньше" в отличие от "more" — "больше"

Чтобы вызвать программу постраничной разбивки, введите в командной строке ее имя, указав затем имя файла для просмотра например `less file.txt` На экране

отобразится первая страница файла. Для постраничного просмотра документа нажи майте пробел. Клавиши <↓> и <↑> позволяют выводить файл построчно Чтобы перейти в начало файла, нажмите <Esc+'<'>, в конец файла – <Esc+'>'> (в случае more это также приведет к завершению программы). Можно даже осуществлять поиск текста введите /, а затем — искомый текст, и программа отобразит результат. Программы постраничной разбивки обладают множеством других функциональных возможностей особенно less Подробнее об этом можно узнать на соответствующих man страницах

Редакторы графического режима

Пользователи Windows и Mac OS привыкли редактировать текст в мощных текстовых процессорах. Но текстовые редакторы, работающие в графическом режиме, есть и во FreeBSD Приведем ряд примеров.

- **Emacs.** Одной из особенностей редактора Emacs является поддержка графических операций, которая улучшается с каждой новой версией редактора Он может работать в отдельном окне, отображая ряд меню, хотя и не полностью интегрируясь в графическую среду. Более тесная интеграция обеспечивается в специальной версии редактора — XEmacs (эта программа может функционировать и в текстовом режиме)
- **NEdit.** Этот редактор (`http: /nedit.org`) ориентирован на пользователей привыкших работать в среде Windows или Mac OS. Он не такой мощный, как Emacs, но его функциональные возможности достаточно велики.
- **gEdit** Это стандартный текстовый редактор графической оболочки GNOME Как редактор он довольно традиционен, в качестве же примера выбран исключительно в силу распространенности.
- **xedit** Это сравнительно простой текстовый редактор, в котором отсутствуют многие функциональные возможности редакторов Emacs или NEdit. Зато он относительно невелик по размеру и нетребователен к ресурсам компьютера

Это тоже далеко не полный список Графические расширения есть и у некоторых редакторов предыдущей категории, но их функциональные возможности ограничены — программа может открыть отдельное окно и отобразить меню, но многие компоненты графической среды (диалоговые окна, шрифты и т.д.) все равно не поддерживаются Просмотреть список доступных текстовых редакторов можно в разделе Editors меню программных пакетов программы sysinstall

Пример: редактирование файла в vi

Как отмечалось выше, редактор vi обладает широкими возможностями, но по современным стандартам его интерфейс слишком непривычен В частности, в нем поддерживаются три разных *режима* функционирования. В зависимости от режима редактор по-разному реагирует на нажатия одних и тех же клавиш поэтому важно знать эту особенность

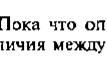
Режимы работы

Три режима работы редактора vi обеспечивают разделение его функциональных возможностей Большинство функций доступно только в одном режиме

- **Командный режим** В этом режиме можно вводить команды, предназначенные для манипулирования текстом или выполнения других действий Названия большинства команд состоят из одного символа Например, команда o встав

ляет строку непосредственно после текущей строки. В командном режиме можно перемещаться по файлу, используя клавиши управления курсором

- **Расширенный режим** В этом режиме выполняются расширенные операции часто связанные с обращением к внешним файлам Именно здесь проявляется гибкость редактора `vi`, так как в расширенном режиме можно запускать внешние программы для выполнения различных операций над файлами (проверка орфографии, форматирование исходного текста программы и т.д.) В этом режиме осуществляется сохранение файла Для перехода в расширенный режим нужно в командном режиме ввести двоеточие а затем — название расширенной команды
- **Режим редактирования** Этот режим соответствует традиционному режиму работы других текстовых редакторов и предназначен для ввода символов в документ Для перехода в командный режим нужно нажать <Esc>



Если вы не знаете точно, в каком режиме находитесь нажмите <Esc> Это переведет программу в командный режим

Пока что описание режимов выглядит несколько абстрактным Чтобы прояснить различия между ними, рассмотрим пример сеанса редактирования в котором используются все три режима

Редактирование файла

Для примера мы выбрали файл `/etc/fstab` в котором устанавливается соответствие между дисковыми разделами и точками монтирования в дереве каталогов FreeBSD На практике приходится добавлять в этот файл описания новых файловых систем и разделов съемных устройств или делать текущие разделы недоступными Поскольку это всего лишь пример, создадим сначала копию файла, чтобы в дальнейшем работать с копией Перейдите в начальный каталог и введите

```
$ cp /etc/fstab  /fstab-copy
```

Не используйте в рассматриваемом примере реальный файл `/etc/fstab`. Это может привести к тому, что система не загрузится. Скопируйте файл `/etc/fstab` в начальный каталог и редактируйте копию Не переходите в режим суперпользователя чтобы случайно не отредактировать исходный файл

Итак, начнем редактирование.

1. Введите команду `vi fstab-copy`, чтобы запустить редактор `vi` и загрузить файл `fstab-copy` в буфер редактирования На рис 5 1 изображено терминальное окно редактора, открытое в среде GNOME В другой графической оболочке вид окна может быть несколько иным. В текстовом режиме никаких меню и полос прокрутки не будет. Символы ~ в левой части экрана используются редактором для обозначения конца файла. В нижней строке отображается статусная информация Конкретный вид файла `/etc/fstab` для нас не важен, поскольку цель примера продемонстрировать, как выполняются операции редактирования
2. Предположим, необходимо удалить одну из записей файла Пусть это будет запись для точки монтирования `/windows` Переместите курсор на эту строку, нажав клавишу <↓> (или `j`) семь раз. Курсор должен оказаться на строке, начинающейся с выражения `/dev/ad0s1`

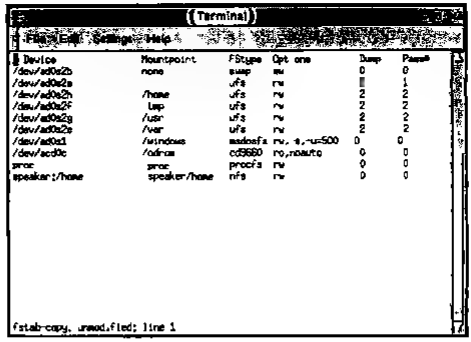


Рис. 5.1 На первый взгляд vi ничем не отличается от других текстовых редакторов его уникальные возможности заключены в поддерживаемых командах

3 Операция удаления текста выполняется в командном режиме, который активизируется при запуске редактора Соответствующая команда называется dd. В результате строка, на которой находится курсор, должна исчезнуть. Если нужно удалить несколько строк, укажите их количество в начале команды, например 3dd (удаление трех строк).

4 Предположим, нужно ввести новую строку вместо только что удаленной. Для начала вставим пустую строку с помощью команды o. Строки, расположенные ниже строки с курсором, должны переместиться на одну позицию вниз, а сам курсор должен оказаться на пустой строке

5 Чтобы ввести строку текста, нужно перейти в режим редактирования В нашем случае этот режим уже включен командой o А вообще для перехода из командного режима в режим редактирования предназначена команда i. Она подготавливает редактор к вставке текста перед курсором Если нужно заменить существующий текст, воспользуйтесь командой R Команда a переводит программу в режим редактирования и перемещает курсор на одну позицию вперед

6 Введите новую строку следующего вида

```
/dev/da0s4      /zip      msdosfs      rw,noauto      0      0
```

7 Нажмите <Esc>, чтобы вернуться в командный режим

8 Теперь создадим другой вариант только что введенной строки, чтобы позволить FreeBSD монтировать Zip-диски с файловой системой FFS, а не FAT Это можно сделать с помощью команды yank (yy), которая копирует текст во внутренний буфер. Собственно команда yy копирует одну строку, а если нужно скопировать несколько строк, укажите их количество в начале команды (например, 4yy).

9 Введите команду p которая вставит текст из буфера обмена Теперь в файле две идентичные строки

10 Переместите курсор на первый пробел после выражения /zip в любой из двух строк.

11 Введите команду R чтобы перейти в режим замены текста

12 Введите **ffs**, чтобы изменить /zip на /zip-ffs

13 Нажмите <Esc>, чтобы вернуться в командный режим и переместите курсор на букву m в поле msdosfs модифицируемой строки

14 Введите команду **R**, чтобы перейти в режим замены текста

15 Введите **ufs** чтобы изменить msdosfs на ufs

На данный момент исходная строка /windows заменена двумя строками, описывающими подключение Zip-дисков Поэкспериментируйте самостоятельно с редактором vi, дублируя, удаляя и редактируя строки, а также создавая новые строки Ниже описаны расширенные возможности редактора

- **Отмена** Редактор vi позволяет отменить последнее выполненное действие. Для этого в командном режиме нужно ввести u Если повторить команду будет отменена сама операция отмены
- **Поиск** Символ косой черты (/), введенный в командном режиме, обозначает начало шаблона поиска. Пусть, например, курсор расположен в начале файла fstab-copy Если ввести **/cd9660** и нажать <Enter>, курсор окажется на сим воле c поля cd9660 девятой строки Знак вопроса (?) задает обратный поиск
- **Замена** Если нужно заменить все экземпляры подстроки в тексте, воспользуйтесь командой глобального поиска Для этого нужно в командном режиме ввести **:%s/***оригинал***/***замена* и нажать <Enter>

Детальные сведения о командах редактора vi приведены в документации Хорошим источником информации является также Web-узел редактора VIM (Vi IMproved). www.vim.org

Сохранение изменений

Осталось выяснить, как сохранить изменения, сделанные в редакторе, и выйти из программы. Для этого существует несколько команд расширенного режима

- **:w** Эта команда записывает сделанные изменения в файл При отсутствии указаний используется имя загруженного файла Если же нужно сохранить изменения в новом файле, задайте его имя, например **:wnew-file.txt**.
- **:e** Если редактор vi вызван отдельно или нужно отредактировать новый файл, воспользуйтесь командой :e, которая загрузит указанный файл, например **:e/etc/printcap**. В каждый момент времени допускается редактирование только одного файла, поэтому нужно либо предварительно сохранить текущий файл, либо отказаться от сделанных изменений, введя команду **:e!**.
- **:r** Эта команда напоминает :e, но добавляет новый файл к существующему
- **:q** Команда выхода из редактора vi

Разрешается объединять две и более команды в одну Распространенный пример — :wq. Эта команда сохраняет текущий файл и завершает работу редактора

Иногда возникает необходимость отказаться от всех сделанных изменений. Для этого предназначен специальный ключ '!' Особенно часто он используется с командами :e и :q. Например команда :q! означает выход из редактора без сохранения изменений.

Пример: редактирование файла в gEdit

Редакторы текстового режима вполне подходят для обработки всех конфигурационных файлов FreeBSD, но большинство пользователей все же предпочитают работать

в графическом режиме, используя соответствующие редакторы Такие редакторы позволяют использовать экранные шрифты, цветовые выделения и т.д.; многие действия удобнее выполнять, выбирая команды из меню. Текстовые редакторы, работающие в графическом режиме, очень разные, но программу gEdit все же можно считать типичным представителем этого класса, к тому же она есть во многих FreeBSD-системах Поэтому мы и остановили на ней свой выбор

Редактирование файла

Как и в предыдущем примере будем редактировать копию файла `/etc/fstab` Перейдите в начальный каталог и введите

```
$ cp /etc/fstab ./fstab-copy
```

Внимание! Не используйте в рассматриваемом примере реальный файл `/etc/fstab`. Это может привести к тому, что система не загрузится Скопируйте файл `/etc/fstab` в начальный каталог и редактируйте копию Не переходите в режим суперпользователя чтобы случайно не отредактировать исходный файл

Запустите графическую оболочку если это еще не сделано и следуйте приведенным ниже инструкциям

1 Откройте терминальное окно и введите команду **`gedit fstab-copy`** чтобы начать редактирование Появится окно редактора gEdit, в котором отображается содержимое файла `fstab-copy` (рис 5 2)

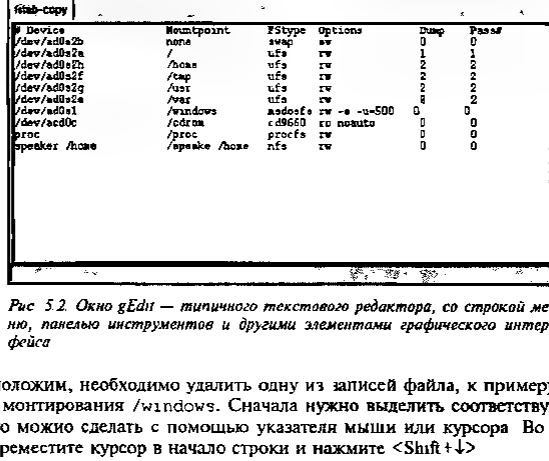


Рис 5 2. Окно gEdit — типичного текстового редактора, со строкой меню, панелью инструментов и другими элементами графического интерфейса

2 Предположим, необходимо удалить одну из записей файла, к примеру запись для точки монтирования `/windows`. Сначала нужно выделить соответствующую строку Это можно сделать с помощью указателя мыши или курсора Во втором случае переместите курсор в начало строки и нажмите <Shift+↓>

3 Удалите выделенную строку щелкнув на кнопке Cut в панели инструментов либо выбрав в меню Edit команду Cut Можно также нажать <Ctrl+X>, <Delete> или <Backspace> В результате строка, на которой находится курсор, должна исчезнуть Если требуется удалить пустую строку, поместите на нее курсор и нажмите <Backspace>

4 Теперь предположим, что нужно добавить новую строку Переместите курсор в нужную позицию, нажмите <Enter>, введите показанный ниже текст и снова нажмите <Enter>.

```
/dev/da0s4      zip      msdosfs      rw,noauto      0      0
```

5 Создадим другой вариант только что введенной строки, чтобы позволить FreeBSD монтировать Zip-диски с файловой системой FFS, а не FAT Выделите строку и скопируйте ее в буфер обмена Для этого нужно щелкнуть на кнопке Copy, выбрать в меню Edit команду Copy или нажать <Ctrl+C>

6 Переместите курсор в начало следующей строки и вставьте строку из буфера щелкнув на кнопке Paste либо выбрав в меню Edit команду Paste Можно также нажать <Ctrl+V>

7 Переместите курсор в конец поля `/zip`

8 Введите `-ffs`, чтобы изменить `/zip` на `/zip-ffs` Затем нажмите четыре раза <Delete>, чтобы удалить лишние пробелы и выровнять оставшиеся колонки

9 Выполните двойной щелчок на слове `msdosfs` новой строки чтобы выделить его

10 Введите `ufs`, чтобы изменить `msdosfs` на `ufs` Нажмите четыре раза пробел для выравнивания последующих колонок

11 Чтобы сохранить изменения, щелкните на кнопке Save или выберите в меню File команду Save Можно также нажать <Ctrl+S> Если нужно сохранить файл под другим именем, воспользуйтесь командой Save As Появится диалоговое окно в котором следует указать новое имя файла

По окончании сеанса редактирования нужно выйти из программы, щелкнув на кнопке Exit, выбрав в меню File команду Exit или нажав <Ctrl+Q> Если программа обнаружит открытые файлы, в которых не сохранены изменения, она выдаст предупреждение.

Дополнительные возможности

Перечислим возможности редактора gEdit, с которыми стоит поэкспериментировать

- **Отмена** Для отмены последней операции нужно выбрать в меню Edit команду Undo или нажать <Ctrl+Z> Это действие можно выполнять несколько раз, отменяя цепочку операций.
- **Поиск**. Если выбрать в меню Edit команду Find или нажать <F6>, появится диалоговое окно Find (рис 5 3) Введите искомую фразу в поле Search for, задайте параметры поиска и щелкните на кнопке Find Чтобы повторить поиск выберите в меню Edit команду Find Again или нажмите <Shift+F6>
- **Замена** При выборе в меню Edit команды Replace или нажатии <F7> появляется диалоговое окно Replace, напоминающее окно Find, но содержащее дополнительное поле Replace with с двумя дополнительными кнопками Replace и Replace all Эти кнопки позволяют заменить, соответственно, первый экземпляр найденной фразы и все экземпляры
- **Параметры** Выберите в меню Settings команду Preferences, чтобы вызвать диалоговое окно настроек программы. Здесь можно задать шрифты, цветовые выделения и другие параметры форматирования текста

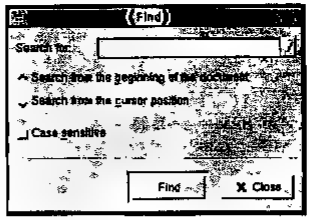


Рис 5 3 В окне Find вводится текст который нужно найти в файле

В отличие от vi программа gEdit может одновременно редактировать несколько файлов Если выбрать в меню File команду Open или нажать <F3>, программа отобразит диалоговое окно выбора файла Для каждого файла в основном окне программы создается отдельная вкладка На рис 5 2 такая вкладка одна: fstab-copy Для переключения между файлами нужно щелкнуть на заголовке вкладки

Важные административные команды

Административные действия зачастую связаны с редактированием текстовых файлов и выполнением команд в текстовом режиме Многие из административных команд будут описаны в последующих главах здесь же рассматриваются важнейшие базовые команды Некоторые из них требуют привилегий суперпользователя. Приведенные команды разбиваются на две основные категории. информационные (выдают сведения о текущем состоянии или конфигурации системы) и манипулирующие (меняют конфигурацию системы).

В этом разделе будет также рассмотрено группирование команд и переадресация ввода-вывода Обычно команды группируются с помощью *каналов*, посредством которых выходные данные одной программы поступают на вход другой Каналы позволяют формировать цепочки команд, выполняемые как одно целое Благодаря *переадресации* программа может направить результаты своей работы в файл или считать входные данные из файла. Все это часто применяется для выполнения административных задач

Команды для получения информации о системе

Следующие команды выдают информацию о работе компьютера Эта информация полезна при поиске неполадок при определении возможностей системы и т д

dmesg

Команда dmesg выводит содержимое буфера сообщений ядра (см. главу 3) Сразу после загрузки компьютера буфер содержит информацию о выявленных аппаратных устройствах, о низкоуровневых аппаратных ошибках и многое другое. В процессе работы системы в буфере накапливаются дополнительные сообщения, связанные с такими событиями, как аппаратные и сетевые тайм-ауты, аномальное поведение программ и т.д. Впрочем, многие из этих событий вполне безобидны и не означают каких-либо серьезных проблем Например, если система сконфигурирована на перевод жесткого диска в режим с пониженным энергопотреблением в буфере могут присутствовать сообщения о тайм-аутах обращения к диску

Синтаксис команды dmesg таков.

```
dmesg [-a] [-M дамп] [-N система]
```

При наличии опции -a создается более подробный отчет, включающий консольные сообщения и сообщения системы Syslog. Опция -M *дамп* позволяет проанализировать *дамп* ядра, т.е. файл, генерируемый при некоторых системных сбоях. Опция -N *система* позволяет просмотреть буфер сообщений ядра другой системы, если у пользователя есть соответствующие полномочия.

df

Команда df сообщает информацию о распределении дискового пространства. Это важно знать при инсталляции новых программ, поскольку может оказаться, что выбранный раздел практически заполнен. Синтаксис команды таков.

```
df [-b | -h | -H | -k | -m | -P] [-ailn] [-t тип] ↵
   [файл | файловая_система ...]
```

Рассмотрим назначение аргументов команды.

- **Опции форматирования.** Опции первой группы задают единицы измерения дискового пространства. -b — 512-байтовые блоки, -h и -H — настраиваемый формат (единицы измерения — килобайты, мегабайты и т.д., по основанию 10 или 2 соответственно), -k — килобайты, -m — мегабайты и -P — вывод в формате POSIX.

На заметку В информатике измерения выполняются по основанию 2, при этом килобайт — это 1024 байт, мегабайт — 1048576 байт и т.д. Производители жестких дисков иногда используют основание 10, считая, что в килобайте 1000 байт, в мегабайте — 1000000 и т.д.

- -a — заставляет команду df сообщать обо всех точках монтирования, даже если некоторые файловые системы смонтированы с параметрами, заставляющими команду df их игнорировать.
- -i — заставляет команду df приводить сведения о свободных *индексных дескрипторах* — дисковых структурах данных, используемых системой для внутренней адресации файлов. Если на диске закончились индексные дескрипторы, на нем нельзя создавать новые файлы, сколько бы свободного пространства ни было. Такая проблема может возникнуть при записи на диск большого количества маленьких файлов.
- -l — заставляет команду сообщать только о локальных файловых системах, расположенных на жестких дисках компьютера, на компакт диске и т.д., но не смонтированных по сети.
- -n — заставляет команду отображать только ту информацию, которая уже хранится в памяти. Обычно команда df опрашивает текущее состояние каждой файловой системы. Если файловая система недоступна, например из-за проблемы в сети, такую информацию получить не удастся.
- -t *тип* — заставляет команду сообщать информацию только о файловых системах указанного типа. Можно задавать список типов, разделенный запятыми. Например, команда df -t ufs,cd9660 выдает информацию о файловых системах FFS и ISO-9660.
- *файл | файловая_система*. Можно указать явный список файловых систем, задав либо имя соответствующего устройства (например, /dev/ad0s2h), либо точку монтирования (/home).

Команда сообщает имя устройства на котором находится файловая система ее объем, объем занятого и свободного пространства процент заполнения и название точки монтирования Вот пример

```
$ df -h
Filesystem     Size    Used   Avail  Capacity  Mounted on
/dev/ad0s2a    124M     62M     51M       54%  /
devfs          1 0K    1.0K      0B      100%  /dev
/dev/ad0s2h    1.8G    486M    1 2G       29%  /home
/dev/ad0s2f    145M     31M    10 M       23%  /tmp
/dev/ad0s2g    1.6G    1.2G    269M       82%  /usr
/dev/ad0s2e    145M     7.4M    126M        6%  /var
/dev/ad0s1     2.0G    697M    1.3G       34%  /windows
procfs         4.0K    4.0K      0B      100%  /proc
speaker:/home  4 5G    1 7G    2.8G       38%  /speaker/home
```

Как показывают результаты работы команды, объем корневой файловой системы (/) — 124 Мбайт, из которых 54% занято Файловая система /home занята на 29% Файловая система /speaker/home смонтирована по сети, о чем свидетельствует имя устройства speaker:/home — это спецификация точки монтирования NFS Файловые системы ,dev и /proc являются *виртуальными*, т.е. им не соответствуют физические разделы. Они создаются динамически для специальных целей Тот факт, что на них доступно 0 байт, не означает наличие проблем А вот файловая система /usr должна побеспокоить пользователя, ведь она заполнена на 82% и для новых программ осталось всего 269 Мбайт

man

Эта команда вызывает справочную систему FreeBSD, ее интерактивное руководство Чтобы получить информацию по интересующей вас теме, введите **man** *тема*, где аргументом может быть имя команды, файла, системного вызова. Команда man поддерживает несколько опций Одна из них -P *программа* — заставляет команду использовать указанную программу постраничной разбивки. Многие предпочитают программу less, а не more, которая задана по умолчанию (установка по умолчанию хранится в переменной среды PAGER)

uptime

Команда uptime не принимает аргументов и выдает информацию о том, как долго компьютер находится во включенном состоянии, сколько пользователей зарегистрировано в системе и какую нагрузку они создают на центральный процессор Результаты работы команды выглядят примерно так.

```
$ uptime
4:38PM up 2 days, 5 mins  5 users, load averages: 0 00, 0 00  0 00
```

Сообщаемое количество пользователей — это в действительности число независимых сеансов регистрации или терминальных сеансов Вполне возможно, что в системе работает всего один пользователь, открывший пять терминальных окон, но по результатам команды uptime это никак нельзя определить

Три числа после фразы load averages — это показатели загруженности центрального процессора с интервалом в 1 минуту, 5 минут и 15 минут Значение 0 00 свидетельствует о низкой загруженности процессора, а значение 1.00 означает, что процессор используется на полную мощность Максимальную загруженность может вызывать единственная программа, выполняющая интенсивные вычисления, или несколько программ, каждая из которых требует от процессора свою "долю внимания" Если показатель средней загруженности превышает 1.0, значит, FreeBSD не может удовлетворить потребности всех программ Программы продолжат выполняться, но не

так быстро как в менее загруженной системе Учтите, что показатель средней загруженности никак не связан с быстродействием процессора Программа, выполняющая интенсивные вычисления, может вызвать среднюю загруженность 1.0 как в системе с процессором Pentium частотой 100 МГц, так и в системе с процессором Pentium 4 частотой 2 ГГц

ps

Эта команда выдает информацию о выполняющихся процессах Она принимает большое число опций, подробное описание которых можно найти на man-странице команды Ниже описаны только наиболее распространенные опции

- `a` вывод информации о процессах всех пользователей (обычно команда `ps` сообщает только о процессах вызвавшего ее пользователя)
- `h` вывод строки с названиями столбцов на каждой странице многостраничных результатов
- `j` — вывод дополнительной информации о каждом процессе
- `l` вывод дополнительной информации о каждом процессе
- `m` — сортировка результатов по объему используемой памяти
- `p` *PID* — вывод информации только для процесса с заданным идентификатором
- `r` — сортировка результатов по степени загруженности процессора.
- `U` *пользователь* — вывод информации о процессах заданного пользователя
- `u` — вывод дополнительной информации о каждом процессе
- `v` — вывод дополнительной информации о каждом процессе
- `w` — создание расширенного, 132-колоночного отчета вместо стандартного 80 колоночного Это может быть удобно, если результаты отображаются в терминальном окне или направляются в файл для последующей обработки
- `x` вывод информации о процессах не связанных с терминалом например демонах

В любой вариант отчета команды включается идентификатор каждого процесса Этот идентификатор можно использовать в целом ряде других команд например `renice` и `kill` (описаны ниже), для изменения работы процесса

О некоторых опциях сказано лишь, что они включают вывод дополнительной информации Какая именно информация, зависит от конкретной опции Это может быть идентификатор родительского процесса, время запуска, различные показатели использования памяти и т д

Особенно полезной является команда `ps aux` Она выдает сведения обо всех выполняющихся процессах, включая имя пользователя, запустившего процесс, процент использования центрального процессора и памяти, а также команду, введенную для запуска процесса Часто результаты выполнения этой команды передаются по каналу утилите `grep`, которая ищет среди них заданную строку Это позволяет обнаружить все экземпляры выполняющейся программы Так в следующем примере выявляются все экземпляры программы `more`

```
$ ps -aux | grep more
rodsmith  7445  0.0  0.2   344   55  p3  R+   5:12PM  0:00.02 grep more
rodsmith  7384  0.0  0.1  1372  244  p2  I+   4:57PM  0:00.08 more
```

На заметку Помимо искомого процесса (в данном случае `more`) находится также процесс самой команды `grep`

top

Команда top является в некотором смысле разновидностью команды ps. Она тоже выдает сводку процессов, которая, однако, обновляется динамически, причем записи сортируются по показателю использования процессора. Это делает команду top чрезвычайно важным средством "отлавливания" неуправляемых процессов, которые заблокировали сами себя в бесконечном цикле и бесцельно потребляют ресурсы центрального процессора. Результаты работы команды представлены на рис. 5.4.

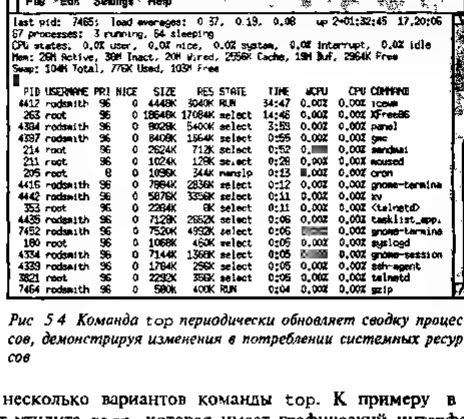


Рис. 5.4. Команда top периодически обновляет сводку процессов, демонстрируя изменения в потреблении системных ресурсов

Существует несколько вариантов команды top. К примеру, в состав оболочки GNOME входит утилита gtop, которая имеет графический интерфейс. Графические варианты команды top функционально аналогичны ее текстовому варианту, но пользователям удобнее работать с ними.

Команды манипулирования системой

Команды, сообщающие информацию о системе, полезны для диагностирования проблем, но чтобы устранять неполадки, нужно иметь возможность менять состояние системы. Ниже описываются команды манипулирования системой и ее процессами.

nice и renice

Каждая программа требует свою долю процессорного времени. Одной из задач ядра FreeBSD является разрешение конфликтов при обращении к центральному процессору. Если, к примеру, два пользователя выполняют ресурсоемкие программы, то кому из них отдавать предпочтение? Обычно FreeBSD распределяет время доступа к процессору между двумя пользователями в соотношении 50/50, чтобы не ущемлять ничьи права. Но иногда такое поведение системы нужно изменить, предоставив одному из заданий более высокий приоритет. Это делается с помощью команды nice, которая позволяет явно присвоить запускаемому процессу код приоритета. С ней связана команда renice, корректирующая приоритет уже выполняющегося процесса.

Синтаксис команды nice таков

```
nice [-число] команда [аргументы]
```

Параметры *команда* и *аргументы* — это запускаемая команда и передаваемый ей список аргументов. Параметр *число* — так называемый *фактор уступчивости*. Чем он выше (до 20), тем ниже приоритет задания, и наоборот: чем он ниже (до -20), тем выше приоритет. Чтобы еще больше все запутать, к числовому аргументу добавляется знак '-'. Таким образом, чтобы понизить приоритет процесса, нужно ввести, казалось бы, отрицательное число:

```
$ nice -15 numbercrunch
```

На самом же деле аргумент -15 трактуется как положительное число. Если действительно нужно указать отрицательное число, следует добавить к нему еще один дефис:

```
# nice --15 numbercrunch
```

Обратите внимание на разные символы приглашения в двух примерах. Это связано с тем, что понизить приоритет запускаемой программы может любой пользователь, а повысить — только суперпользователь. Если числовой аргумент не указан, команда nice подразумевает фактор уступчивости 10.

Необходимость в команде nice возникает не так часто, но она чрезвычайно удобна для контроля низкоприоритетных процессов, потребляющих слишком много ресурсов центрального процессора. Например, популярная программа SETI@Home (http://setiathome.ssl.berkeley.edu) занимает очень много процессорного времени, поэтому ее обычно запускают с очень низким приоритетом (значение nice равно 20), чтобы она запускалась только в отсутствие других пользовательских процессов.

Команда renice расширяет функциональные возможности команды nice, позволяя менять приоритет выполняющегося процесса. Ее синтаксис таков:

```
renice фактор_уступчивости [[-p] PID] [[-g] группа] [[-u] пользователь]
```

Можно задать идентификатор процесса, группу пользователей - владельцев процесса или имя пользователя, запустившего процесс. Вот как, к примеру, можно повысить приоритет всех процессов пользователя ajones:

```
# renice -1 ajones
```

Учтите, что, в отличие от команды nice, к числовому аргументу не добавляется дефис. Следовательно, увеличению приоритета соответствует отрицательный аргумент. Чтобы не путать идентификаторы процессов, названия групп и имена пользователей, можно явно указывать опции -p, -g и -u. Повышать приоритет процесса разрешено только суперпользователю. Остальные пользователи могут выполнять команду renice только для собственных процессов.

Рядовой пользователь, понизивший приоритет процесса с помощью команды nice или renice, уже не сможет восстановить исходное значение приоритета.

Если вы обнаружили, что работа системы замедлилась из-за нескольких программ, понизьте их приоритет с помощью команды renice. Такое действие эффективно, если программа потребляет много ресурсов процессора, выполняя, к примеру, операции моделирования. Для программ, осуществляющих интенсивный ввод-вывод, это вряд ли оправданно.

kill и killall

Команды nice и renice полезны с точки зрения эффективного распределения ресурсов центрального процессора между низкоприоритетными и высокоприоритетны

ми заданиями. Но иногда этого недостаточно. Бывает, что процессы "зависают", т. е. блокируют себя в цикле ожидания. Такие процессы все равно могут отнимать немало времени процессора, причем без всякой пользы. Случается, что программа перестает реагировать на команды пользователя, и нужно как-то выйти из нее. В этом случае на помощь приходят команды `kill` и `killall`. Они могут завершить работу заданного процесса, даже если он завис и не реагирует на запросы.

Синтаксис команды `kill` таков:

```
kill [-s] название_сигнала | номер_сигнала] PID
```

В простейшем случае все необязательные аргументы опускают, указывая только идентификатор процесса, который можно узнать с помощью команды `ps` или `top`. Вот как найти некую программу `xv` и уничтожить ее:

```
# ps ax | grep xv
 7672  p2  RN+    0:00.02 grep xv
 4442  p5  I      0:13.40 xv
# kill 4442
```

Команда `kill` посылает процессу *сигнал* – специальный код, являющийся средством межзадачного взаимодействия. По умолчанию посылается сигнал номер 15, называющийся TERM (от "terminate" — завершить). Большинство процессов реагирует на этот сигнал, завершая работу стандартным образом. К сожалению, если процесс завис, он не реагирует даже на сигнал TERM. На этот случай предусмотрен сигнал номер 9, KILL. Например, если ввести `kill -s 9 4442`, то сигнал KILL будет послан процессу с идентификатором 4442. В действительности этот сигнал передается не процессу, а непосредственно ядру, заставляя его немедленно уничтожить процесс. У процесса не будет даже возможности закрыть открытые файлы и выполнить другие завершающие процедуры. Некоторые из наиболее распространенных сигналов перечислены в табл. 5.1. Полный их список можно получить, выполнив команду `kill -l`.

Таблица 5.1. Распространенные сигналы и их назначение

Название сигнала	Номер сигнала	Назначение
HUP	1	"Отбой" разрыв связи
INT	2	Прерывание
QUIT	3	Выход
ABRT	6	Аварийное завершение
KILL	9	Безусловное уничтожение
ALRM	14	Сигнал от таймера
TERM	15	Корректное завершение

Команда `killall` служит аналогичным целям, но уничтожает все процессы, соответствующие заданному имени. Например, можно не вводить команды **`ps ax | grep xv`** и **`kill 4442`**, как в показанном выше примере, а сразу ввести **`killall xv`**, что приведет к тому же результату. Проблема заключается в том, что можно непреднамеренно уничтожить полезные процессы. Если, допустим, два пользователя запустили программу `xv`, то команда `killall xv`, выполненная от имени суперпользователя, уничтожит оба экземпляра программы.

Внимание! В некоторых UNIX-системах используется другая версия команды `killall`. Она уничтожает все процессы, запущенные текущим пользователем, за исключением тех, которые являются предками самого процесса `killall`. Подробности можно узнать на man-странице комманды.

Команды kill и killall доступны как суперпользователю так и рядовым пользователям Однако последние могут посылать сигналы только собственным процессам

Переадресация ввода-вывода

Средства переадресации ввода вывода чрезвычайно важны не только для администраторов, но и для рядовых пользователей. Они позволяют менять каналы ввода вывода программ, соединяя их с файлами, а не с клавиатурой или экраном

Чтобы понять суть переадресации, нужно прежде всего уяснить, что большинство программ может принимать и отображать данные через разные источники (называемые *потоками*) Обычно входной поток программы связан с клавиатурой компьютера, за которым работает пользователь, а выходной поток направлен на экран или в терминальное окно Это так называемые *стандартный входной* и *стандартный выходной* потоки (обозначаются как stdin и stdout) Есть еще второй выходной поток, называемый *потоком ошибок* (stderr) Через него передаются важные сообщения об ошибках Обычно он направлен туда же, куда и поток stdout, но это два разных потока.

Операторы переадресации позволяют связать входные и выходные потоки с другими источниками или целевыми устройствами Например, результаты работы программы можно направить не на экран, а в файл для последующего редактирования Переадресация потока stdout осуществляется с помощью оператора >, после которого указывается имя файла Следующая команда сохраняет длинный листинг содержимого каталога в файле

```
$ ls  l > listing txt
```

Полученный файл можно отобразить с помощью программы постраничного просмотра, такой как less, загрузить в текстовый редактор или подвергнуть другой обработке. Некоторые программы больше работают с потоком stderr, а не stdout. Идея заключается в том, что пользователь должен немедленно прочесть сообщение об ошибке, а не дожидаться вывода остальных результатов К сожалению, это часто приводит к путанице в выходных данных, вследствие чего сообщения об ошибках часто перенаправляют в отдельный файл Это можно сделать с помощью оператора 2> На пример, если программа bigerr генерирует большое число сообщений об ошибках переадресуйте поток stderr следующим образом

```
$ bigerr 2> bigerr-error-output.txt
```

Чтобы переадресовать оба потока, введите такую команду

```
$ bigerr > bigerr-output.txt 2> bigerr-error-output txt
```

Если целевой файл существует, он будет перезаписан Чтобы добавлять данные в файл, не перезаписывая его, воспользуйтесь оператором >>

```
$ ls  l >> listing txt
```

Многие программы принимают данные, вводимые с клавиатуры Если входная информация четко структурирована, то можно предварительно занести ее в файл, передав его затем на вход программы Оператор переадресации входного потока — < В следующем примере сценарию getinfo передаются данные, содержащиеся в файле address.txt

```
$ getinfo < address.txt
```

Разрешается одновременно переадресовывать входной и выходной потоки

```
$ getinfo < address txt > get-out.txt
```

В данном случае сценарий getinfo читает данные из файла address txt и осуществляет запись в файл get-out.txt

Каналы

Концепция каналов, или программных конвейеров, как их иногда называют, тесно связана с переадресацией ввода-вывода. По сути, канал — это связующее звено между двумя программами, позволяющее одной программе передавать свои выходные данные на вход другой. К каналу могут подключаться дополнительные "звенья". Тогда вторая программа будет передавать выходные данные третьей и т.д. Таким способом часто соединяют небольшие, несложные программы, чтобы совместно они решали более сложную задачу.

Каналы уже использовались ранее, в примерах с командами `ps` и `grep`. Команда `ps` способна сгенерировать довольно большой отчет — по одной строке на каждый найденный процесс, а их могут быть десятки и сотни даже в относительно небольшой системе. Пользователя же интересует всего несколько строк, поэтому направление результатов по каналу команде `grep` — самый простой выход из ситуации. Эта команда ищет во входном потоке строки, содержащие заданный шаблон. В случае команды `ps` таковым будет имя процесса. Оператор канальной переадресации записывается в виде вертикальной черты.

```
$ ps  aux | grep xv
```

Эта "объединенная" команда ищет в выводе команды `ps` строки, содержащие слово `xv`, и отображает их на экране. Такой прием часто применяют для "просеивания" длинных листингов. Естественно, применение каналов этим не ограничивается. Можно, например, перенаправить список файлов, найденный командой `find`, утилите резервного копирования или передать длинный листинг программе постраничной разбивки.

```
$ ps -aux | less
```

О командах `grep` и `find` рассказывается в главе 8.

Графические административные утилиты

В этой главе рассматриваются преимущественно утилиты, работающие в текстовом режиме, но во FreeBSD есть и графические административные программы. Так традиционно сложилось, что администраторы FreeBSD реже пользуются графическими средствами, поскольку стандартная конфигурация системы ориентирована в большей степени на текстовый режим. Графические средства удобны для администраторов-новичков, а также для пользователей, которые хотят выполнять определенные административные задачи. В этом случае графический интерфейс позволяет быстрее находить нужное решение. Но возможности графических административных средств обычно более ограничены, чем у их текстовых аналогов, поэтому они могут не вполне подходить для повседневного администрирования системы.

Ниже будут вкратце рассмотрены основные возможности некоторых административных программ.

- **`sysinstall`**. Это, вообще говоря, не графическая программа, но она является стандартным компонентом FreeBSD и выполняет многие функции своих графических аналогов. Именно она контролирует процесс инсталляции системы, как описывалось в главе 2. Для запуска программы нужно ввести ее имя в командной строке. Программа позволяет инсталлировать дополнительные программные пакеты, конфигурировать X среду, задавать часовой пояс и многое другое. Мы будем периодически ссылаться на нее в различных главах.

- **Webmin**. Эта программа доступна на Web-узле `http://www.webmin.com`. Она представляет собой Web-ориентированный пакет административных средств для FreeBSD и многих других разновидностей UNIX. Круг поддерживаемых административных задач довольно широк. К программе подключается множество модулей, управляющих конфигурированием серверов и других системных компонентов.
- **SWAT**. Программа SWAT (Samba Web Administration Tool) входит в состав пакета Samba (глава 18). Как и Webmin, она имеет Web-ориентированный интерфейс. Программа взаимодействует только с сервером Samba, зато позволяет контролировать все его функциональные возможности и быстро настраивать довольно сложные конфигурации.
- **CUPS**. CUPS (Common Unix Printing System — общая система печати в UNIX) является альтернативой стандартным средствам печати во FreeBSD, которые описаны в главе 9. Мы включили CUPS в этот перечень, потому что в состав системы входит еще одна Web-ориентированная административная утилита, управляющая конфигурацией принтера. Более подробную информацию можно найти на Web-узле `www.cups.org`.

Программы с Web-ориентированным интерфейсом пользуются популярностью у администраторов. Они позволяют администрировать FreeBSD с любого компьютера, где есть Web-броузер, независимо от операционной системы. Но пользоваться такими программами для выхода за пределы локальной сети слишком опасно, так как злоумышленнику ничего не стоит перехватить пароль администратора. Если администрирование осуществляется с центрального сервера, то в случае его взлома под угрозой оказывается вся сеть. Даже в локальной сети администрирование через Web-броузер небезопасно, если внутренняя безопасность сети нарушена. Короче говоря, пользуйтесь этими средствами с осторожностью.

Резюме

Системное администрирование включает в себя запуск специальных программ для выполнения административных задач и редактирование текстовых файлов, содержащих конфигурационные установки системы. Важно понимать, какие функции выполняют соответствующие команды и какая информация хранится в конфигурационных файлах. Нужно также уметь пользоваться текстовыми редакторами, доступными во FreeBSD.

Эта глава является лишь введением в системное администрирование. Большая часть информации, которую должен знать администратор FreeBSD, приведена в последующих главах.

Полный справочник по

Глава 6

Запуск системы и управляющие процессы

Одна из наиболее важных административных задач — контроль процессов загрузки системы и регистрации пользователей в ней. Если что-то сконфигурировано неправильно, система может оказаться недоступной для пользователей и придется использовать загрузочную дискету. В предыдущих главах уже рассматривалась начальная загрузка FreeBSD. В настоящей главе мы еще раз обратимся к этой теме и поговорим о том, как происходит регистрация пользователей в системе.

Даже когда никто из пользователей не зарегистрирован в системе, в ней все равно выполняются определенные программы. Многие из них вызываются из сценариев запуска системы, но есть еще один класс программ, которые обрабатываются особым образом. Это запланированные задания, контролируемые демоном cron. В конце главы будет показано, как планировать выполнение системных и пользовательских заданий.

В этой главе описывается множество программ и демонов, схема связей между которыми представлена на рис. 6.1. На схеме изображены далеко не все процессы, выполняемые во FreeBSD, но она полезна для понимания того, как осуществляется контроль процессов в системе и в какой примерно последовательности они запускаются.

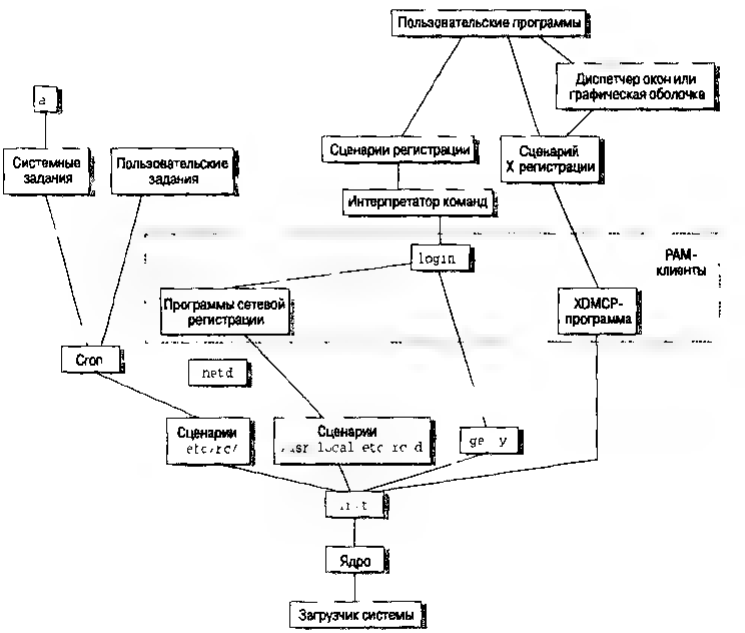


Рис. 6.1. Схема взаимосвязи управляющих процессов и сценариев FreeBSD

Системное ядро

Ядро – это "сердце" операционной системы. Оно выполняет множество функций, которые настолько важны, что разделы, посвященные ядру, содержатся в нескольких главах книги. Ниже вкратце рассказано о том, как происходит инициализация ядра и какую роль играет ядро стандартной системы.

Загрузка ядра

Загрузка ядра подробно рассматривалась в главе 4. Вспомним еще раз, как все происходит. Ядро FreeBSD оказывается в памяти "благодаря" системному загрузчику. Если говорить конкретнее, то сначала загрузчик этапа 0 вызывает загрузчик этапа 1 из первичного раздела FreeBSD. Тот в качестве встроенного компонента запускает загрузчик этапа 2. Именно загрузчик этапов 1–2 отвечает за инициализацию ядра. В стандартной конфигурации ядро читается из файла `/boot/kernel/kernel`, но эту установку можно изменить с помощью команды `boot` вторичного загрузчика, как описывалось в главе 4.

Оказавшись в памяти, ядро начинает контролировать работу компьютера. Любая программа, которая попытается в обход ядра напрямую обратиться к аппаратному устройству, потерпит неудачу. Вместо этого ядро реализует высокоуровневые интерфейсы доступа к оборудованию компьютера.

Влияние ядра на последующие этапы загрузки системы

Первая программа, запускаемая ядром, – это демон `init`, управляющий инициализацией системы. Он использует интерпретатор sh для вызова целой группы сценариев запуска. Ядро не контролирует эти этапы, но его присутствие важно по двум причинам. Во-первых, ядро определяет, какая программа управляет процедурой запуска системы. В принципе, можно заставить ядро запускать какую-нибудь другую программу вместо демона `init`, поменяв тем самым схему работы системы. Во-вторых, ядро реализует вспомогательные функции, используемые демоном `init`, управляющими сценариями и программами, которые вызываются из этих сценариев. Аналогичные функции доступны всем программам, выполняющимся в работающей системе.

Роль ядра в управлении работающей системой

Пользователи редко вспоминают о присутствии ядра, но именно оно контролирует все действия пользователей. Службы ядра выполняются в так называемом *режиме ядра*. Эти службы имеют привилегированный доступ к аппаратным устройствам и другим процессам ядра. В отличие от этого программы *пользовательского режима* получают косвенный доступ к устройствам через службы ядра. Нужно понимать, что даже программы, запускаемые пользователем `root`, выполняются в пользовательском режиме.

Ниже перечислены ключевые механизмы ядра.

- **Аппаратные интерфейсы.** Ядро включает драйверы большинства аппаратных компонентов. Если программе нужно записать данные в последовательный порт, прочитать данные, введенные с клавиатуры, отобразить данные на экране или обратиться к другому устройству, она делает это через посредство ядра. Некоторые обращения могут буферизоваться промежуточными службами на

пример файловыми системами или XFree86, но рано или поздно в игру вступит ядро. На практике доступ к большинству аппаратных интерфейсов осуществляется через *файлы устройств*. Это специальные файлы, находящиеся в каталоге `/dev`.

На заметку

Некоторые устройства, например модемы и принтеры, не поддерживаются ядром напрямую. Вместо этого они контролируются пользовательскими программами, которые тем не менее обращаются к ним через устройства, контролируемые ядром, в частности последовательные и параллельные порты. Система XFree86 заслуживает особого упоминания из-за ее необычного статуса. Она выполняется в пользовательском режиме, но при этом имеет прямой доступ к ресурсам графической платы.

- **Контроль доступа к памяти**. Все программы обращаются к памяти, и, чтобы программы не перезаписывали данные друг друга, ядро отвечает за корректное выделение блоков памяти. При этом каждая программа выполняется как бы сама по себе. Ни одна программа не может обратиться к памяти другой программы. В некоторых старых операционных системах, таких как Mac OS Classic и Windows 3.1, применялись более примитивные методы выделения памяти, и это была одна из причин того, почему прикладные программы часто вызывали сбой операционной системы.
- **Контроль доступа к центральному процессору**. FreeBSD является операционной системой с *вытесняющей, или приоритетной, многозадачностью*. Это означает, что ядро контролирует, какую долю процессорного времени получает каждый процесс. При *кооперативной многозадачности* процессы работают с процессором столько, сколько им нужно, после чего передают управление другим процессам. Это менее удачный режим, поскольку какая-нибудь программа может захватить процессор на непропорционально длительное время. Во FreeBSD приоритеты процессов, определяющие, какую долю процессорного времени они получают, контролируются командами `nice` и `renice` (глава 5).
- **Файловые системы**. Каждый раз, когда происходит обращение к файлу, ядро выступает в качестве посредника. Программа направляет ядру запросы на чтение данных из файла и запись данных в файл. Следовательно, ядро должно понимать структуру всех файловых систем, с которыми ведется работа. Эта тема рассматривается в главах 4 и 7. Файловые системы взаимодействуют с аппаратными интерфейсами, контролирующими доступ к дисковым устройствам. Таким образом, в доступе к файлам участвуют два компонента ядра: сама файловая система и аппаратный интерфейс того устройства, на котором находится файловая система.
- **Сетевые функции**. Помимо контроля сетевого оборудования ядро реализует сетевые функции более высокого уровня. В частности, оно осуществляет поддержку *стека TCP/IP* — группы протоколов, упрощающих пользовательским программам доступ к сети. Во FreeBSD есть множество программ, позволяющих контролировать сетевые компоненты ядра и взаимодействовать с ними. Об этих программах рассказывается в частях III и IV, а также в ряде других глав.

Как видите, ядро является чрезвычайно важным компонентом любой операционной системы. Ядро FreeBSD напоминает ядра других BSD-систем с открытым исходным кодом (OpenBSD и NetBSD) и отличается от ядра Linux. Вот почему FreeBSD и Linux — совершенно разные системы, несмотря на то что они поддерживают много одинаковых высокоуровневых программ, таких как XFree86 и Emacs.

Ядро FreeBSD тщательно протестировано и очень устойчиво В этом секрет надежности FreeBSD Даже если пользовательская программа ведет себя некорректно, это вряд ли скажется на работе ядра или других программ. Исключением является случай, когда программы взаимодействуют друг с другом и, в зависимости от привилегий и параметров доступа, могут повлиять друг на друга Например, программа, запущенная пользователем root, может уничтожить или как-то повлиять на программу запущенную другим пользователем, но даже программы суперпользователя должны обращаться к ядру для доступа к аппаратным устройства компьютера, выделения памяти и т д. Подобная изоляция чрезвычайно важна для обеспечения стабильности системы, в которой одновременно могут работать сотни пользователей

Настройка сценариев запуска системы

О сценариях запуска уже говорилось в главе 5. Во FreeBSD эти сценарии бывают двух видов

- **Системные сценарии** Эти сценарии находятся в каталоге /etc, а их имена начинаются с префикса rc Сценарий /etc/rc является главным среди них, и демон init запускает его первым Сценарии запуска системы формируют стандартную рабочую конфигурацию FreeBSD и благодаря им начинает работать множество стандартных серверов
- **Локальные сценарии** Находятся в каталоге /usr/local/etc/rc.d и обеспечивают запуск вспомогательных серверов и других программ, добавляемых к стандартной конфигурации системы Эти сценарии должны реагировать как минимум на две команды — start и stop, означающие запуск и останов соответствующих программ (В ходе загрузки системы сценарии вызываются автоматически, впоследствии их можно запускать и останавливать вручную)

Если нужно изменить процедуру загрузки системы модифицируйте либо системные, либо локальные сценарии запуска По возможности старайтесь работать только с локальными сценариями. Такой подход удобен, когда требуется добавить поддержку нового сервера или программного пакета, запускаемого на этапе загрузки системы Это позволяет в случае необходимости возвращаться к стандартной конфигурации

Редактирование системных сценариев

Сценарии запуска системы следует редактировать с осторожностью, поскольку даже небольшая ошибка может привести к тому, что система не загрузится. К этому следует прибегать лишь в крайнем случае, когда без непосредственного редактирования сценариев не обойтись. Но есть два исключения из правила.

Во-первых, можно вполне безбоязненно редактировать сценарий /etc/rc.conf В нем определяется ряд переменных, используемых другими сценариями для контроля процесса загрузки Названия этих переменных вполне очевидны hostname sendmail_enable и sshd_enable.

На заметку Значения, задаваемые в файле /etc/rc conf перекрывают установки файла /etc/defaults rc.conf

Во вторых, можно отредактировать файл /etc/rc.local чтобы заставить систему запускать дополнительные серверы В стандартной конфигурации этот сценарий не существует, но сценарий /etc/rc ищет его и запускает, если он присутствует В листинге 6 1 показан фрагмент сценария /etc/rc.local, в котором запускается сервер

/usr/local/bin/servera с аргументом -a и выполняется сценарий /usr/local/bin/startcheck

Листинг 6.1. Простейший пример сценария /etc/rc.local

```
#!/bin/sh
/usr/local/bin/servera -a &
/usr/local/bin/startcheck
```

Сценарий начинается со строки #!/bin/sh. Она означает, что файл является *сценарием интерпретатора команд*, т.е. программой, выполняемой с помощью интерпретатора (в данном случае /bin/sh). Для большинства сценариев устанавливают бит выполнения (это можно сделать с помощью команды chmod a+x), но для данного сценария это необязательно. Последующие записи файла выглядят так, как если бы их вводили в командной строке. Оператор & в строке вызова сервера servera означает, что программа запускается в фоновом режиме. Необходимость в этом операторе определяется особенностями сервера. Некоторые серверы переходят в фоновый режим даже в отсутствие явного указания со стороны пользователя.

Сценарий /etc/rc.local удобен для запуска локальных серверов, однако такой метод считается устаревшим, т.е. использовать сценарий не рекомендуется. В будущих версиях FreeBSD поддержка сценария может прекратиться. Более правильный метод — использовать локальные сценарии запуска, описываемые ниже.

Создание новых локальных сценариев

Вместо одного большого сценария /etc/rc.local лучше создать несколько сценариев в каталоге /usr/local/etc/rc.d. Они обычно сложнее сценария /etc/rc.local из-за необходимости обрабатывать обязательные аргументы start и stop. Пример такого сценария показан в листинге 6.2.

Листинг 6.2. Образец локального сценария из каталога /usr/local/etc/rc.d

```
#!/bin/sh

case "$1" in
start)
    /usr/local/bin/servera -a
stop)
    /usr/bin/killall servera
```

Это очень простой сценарий. Если получен аргумент start, запускается сервер servera. Если же получен аргумент stop, выполняется команда killall, уничтожающая все экземпляры сервера. Более сложные сценарии записывают идентификатор процесса в файл, чтобы при получении команды stop уничтожался только тот экземпляр, который был непосредственно запущен сценарием. Можно также проверять наличие программного файла перед запуском сервера и т.д. Для локальных сценариев нужно устанавливать бит выполнения с помощью команды chmod a+x /usr/local/etc/rc.d/*имя_сценария*.

Многие серверные пакеты, скомпилированные для FreeBSD, включают готовые локальные сценарии, предназначенные для запуска и останова сервера, чтобы не нужно было создавать их вручную. Существующий сценарий можно использовать как образец для создания нового сценария, хотя в некоторых сценариях используются довольно сложные методики программирования.

Особенности регистрации в системе

Когда система загружена, появляется приглашение на регистрацию. Дальнейшие события зависят от того, какой метод входа в систему выбран, хотя некоторые моменты являются общими для всех методов. В стандартной конфигурации FreeBSD, введя правильный пароль, пользователь беспрепятственно входит в систему. В этом разделе описывается, как изменить стандартные настройки всех учетных записей и порядок регистрации отдельных пользователей. Будет также рассказано о том, что происходит при входе пользователя в систему.

Методы регистрации

FreeBSD поддерживает несколько методов регистрации в системе. Несмотря на внешние различия, в их основе лежат одинаковые средства аутентификации, знание которых поможет правильно организовать доступ в систему.

Консольная регистрация в текстовом режиме

Такой метод регистрации описывался в главе 3. По приглашению `login:` пользователь должен ввести свое регистрационное имя, а затем, в ответ на приглашение `password:`, — пароль. После этого запускается интерпретатор команд. Это основной метод входа в систему в стандартной инсталляции FreeBSD.

Описанный процесс контролируется с помощью файла `/etc/ttys`, который содержит группу строк следующего вида:

```
ttyv0   "/usr/libexec/getty Pc"         cons25  on  secure
```

Эта строка связывает программу `/usr/libexec/getty` с устройством `ttyv0`, которому во FreeBSD соответствует первая виртуальная консоль. Последующие строки связывают ту же самую программу с консолями `ttyv1`–`ttyv7`. Для доступа к этим консолям нужно нажать <Alt+F*n*>, где *n* — номер консоли от 1 до 8 (клавише <F1> соответствует консоль `ttyv0` и т.д.). Таким образом, в системе можно регистрироваться несколько раз.

Программа `getty` подключается к консоли и запускает программу `login`, которая управляет процессом регистрации в системе. Именно она отображает приглашения `login:` и `password:`, обрабатывает вводимые пользователем данные и вызывает *подключаемые модули аутентификации* (Pluggable Authentication Module — PAM). Если пользователь прошел аутентификацию, программа `login` обращается к файлу `/etc/passwd` и выясняет, какой интерпретатор команд закреплен за пользователем, после чего запускает этот интерпретатор, соединяя его с выбранной консолью. С этого момента пользователь считается зарегистрированным в системе, хотя интерпретатор команд может выполнять дополнительные сценарии, устанавливая заданные пользователем переменные среды.

Консольная и дистанционная регистрация в графическом режиме

Регистрацией в графическом режиме управляют программы, работающие по протоколу *XDMCP (X Display Manager Control Protocol)*. Такая программа либо работает в рамках X-сеанса, либо принимает запросы на подключение от дистанционных X-серверов. В любом случае программа отображает графическое приглашение на регистрацию в системе. Когда пользователь вводит свое имя и пароль, управляющая программа передает эту информацию модулям PAM. Если аутентификация пользователя прошла успешно, программа запускает закрепленную за пользователем графическую оболочку.

Во FreeBSD есть несколько XDMCP-программ, включая XDM — первую из этой серии программ, GDM, являющуюся частью пакета GNOME, и KDM, которая входит в состав KDE. Эти программы вызываются из файла `/etc/ttys`, как и описанная выше программа `getty`. Об их конфигурировании подробнее рассказывается в главе 21.

Если требуется подключить такую программу, найдите в файле `/etc/ttys` следующую строку:

```
ttyv8   /usr/X11R6/bin/xdm  nodaemon"  xterm  off  secure
```

Поменяйте `off` на `on`, а затем остановите X-сервер, если он запущен, и введите команду `kill -HUP 1` от имени суперпользователя. Это заставит демон `init` (его идентификатор процесса всегда равен 1) повторно прочесть свои конфигурационные файлы, в том числе и `/etc/ttys`. После небольшой паузы будет запущен X-сервер и на экране появится приглашение к регистрации, выдаваемое программой XDM (рис. 6.2). Если используется программа GDM или KDM, приглашение будет выглядеть немного иначе.

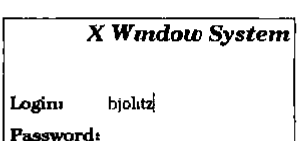


Рис. 6.2. Приглашение на вход в систему в графическом режиме

Дистанционная регистрация в текстовом режиме

Если система сконфигурирована на прием дистанционных запросов на регистрацию в текстовом режиме, то управляющие серверы могут работать подобно программе `getty`. В частности, сервер Telnet ожидает сетевых подключений. Когда поступает сетевой запрос, сервер запускает программу `login`, которая работает так, будто имеет дело с виртуальной консолью, вот только данные передаются по сети, без участия локальных клавиатуры и монитора.

По-другому реализован протокол *SSH (Secure Shell)*. Вместо программы `login` сервер SSH обращается непосредственно к модулям PAM либо к собственным механизмам аутентификации в обход модулей PAM. Это обеспечивает большую гибкость. По завершении этапа аутентификации сервер SSH соединяет интерпретатор команд пользователя с дистанционным клиентом SSH, предоставляя пользователю доступ в систему.

Программа `rlogin` — одно из старейших средств дистанционной регистрации в UNIX. По умолчанию она аутентифицирует пользователя на основании IP-адреса его компьютера, а модули PAM не используются. Когда поступает запрос на регистрацию в системе, программа сверяет IP-адрес дистанционной системы с адресами, записанными в файле `/etc/hosts.allow`, `/etc/hosts.equiv` или пользовательском файле `.rhosts`. Если адрес найден в одном из этих файлов и у пользователя есть учетная запись на сервере, ему предоставляется доступ к интерпретатору команд.

На заметку Серверы Telnet и SSH рассматриваются в главе 21. Программа `rlogin` считается небезопасной по современным стандартам, поэтому ее лучше избегать.

Все службы дистанционной регистрации контролируются серверами, которые нужно как-то запускать. Иногда это делают локальные сценарии запуска, а в осталь-

ных случаях — демон inetd, описанный в главе 16. Сам демон вызывается из системных управляющих сценариев (см. рис. 6.1).

Во FreeBSD поддерживается регистрация через последовательные порты. В былые времена к последовательным портам большинства UNIX-систем были подключены *терминалы ввода-вывода*, которые представляли собой клавиатуру и монитор без системного блока. Такая конфигурация до сих пор поддерживается и позволяет нескольким пользователям получать одновременный доступ в систему. В качестве управляющей станции может выступать обычный компьютер, оснащенный терминальной программой. К последовательному порту можно подключить и модем, разрешая дистанционный коммутируемый доступ в систему. Все эти конфигурации тоже настраиваются в файле /etc/ttys. К примеру, следующая строка заставляет систему принимать запросы на регистрацию в системе через модем:

```
ttyd0    /usr/libexec/getty std.9600"   dialup   on   secure
```

В этом случае демон init будет подключать программу getty к устройству /dev/ttyd0, которое связано с первым последовательным портом (COM1 в Windows). Выражение std.9600 означает, что используется модем со скоростью 9600 бит/с. Ключевое слово dialup идентифицирует терминал как коммутируемое модемное соединение. Типы терминалов определены в базе данных /etc/termcap.

Специальные серверы

Некоторые серверы предоставляют своим пользователям дистанционный доступ в систему. Это, в частности, делают серверы FTP и Samba, позволяющие пользователям получать доступ к файлам. Еще более ограниченные возможности предоставляются серверами доставки почты. Эти серверы могут запрашивать у пользователя пароль для входа в систему. Некоторые серверы взаимодействуют с модулями PAM. В любом случае процесс регистрации в системе зависит от сервера. Иногда настройки серверов содержатся в общесистемных конфигурационных файлах, иногда — в пользовательских файлах. О серверах рассказывается в части IV.

Аутентификация пользователей с помощью модулей PAM

Модули PAM — важное средство централизованной аутентификации во FreeBSD. До появления в UNIX-системах модулей PAM программа регистрации считывала зашифрованный пароль из файла паролей, шифровала пароль, введенный пользователем, и сравнивала результаты. У этого метода был ряд недостатков. Например, в каждой программе приходилось реализовывать дополнительные проверки. Если менялся способ шифрования файла паролей, программы приходилось переписывать. С появлением модулей PAM эти проблемы исчезли. Специальные методы аутентификации, изменения в формате паролей и другие настройки теперь реализуются один раз, в коде соответствующего модуля. Программы, работающие с этим модулем (называемые *PAM-клиентами*), могут даже не "знать" об этих изменениях.

У модулей PAM свои конфигурационные файлы. Во FreeBSD 5.0 они находятся в каталоге /etc/pam.d, причем каждый файл называется так же, как и программа, которая работает с модулями PAM. В более ранних версиях FreeBSD использовался единый конфигурационный файл /etc/pam.conf. Его формат аналогичен формату нынешних файлов за одним исключением, о котором будет сказано ниже.

Модули PAM реализуют четыре типа аутентификации:

- **Аутентификация пользователей**. Модуль аутентификации разрешает или запрещает доступ в систему на основании имени пользователя и пароля. Такой модуль имеет тип auth.

- **Управление учетными записями.** Модуль сообщает серверу, является ли учетная запись пользователя корректной. Это очень важно для протоколов дистанционной регистрации, которые часто не поддерживают аутентификацию. Например, модуль может запретить доступ к учетной записи в зависимости от времени суток или если срок действия пользовательского пароля истек. Такой модуль имеет тип `account`.
- **Управление паролями.** Некоторые модули позволяют программам менять пользовательские пароли. Обычно такая возможность не используется в процессе регистрации пользователя, но она полезна для утилит наподобие `passwd` (описана в главе 10). Такой модуль имеет тип `password`.
- **Управление сеансами.** Модули типа `session` позволяют программам управлять сеансами доступа пользователей в систему.

Необходимость в изменении конфигурации модулей PAM возникает редко. Если требуется модифицировать порядок аутентификации для того или иного метода регистрации в системе, нужно отредактировать соответствующий конфигурационный файл, добавив, удалив или изменив строку вызова определенного модуля. Модуль PAM представляет собой библиотеку функций, реализующих определенный алгоритм аутентификации. Строка вызова модуля имеет следующий формат:

[название_службы] тип_модуля управляющий_флаг модуль [параметры]

В первом, опциональном, поле задается название службы, например `login` или `su`. Это поле присутствует только в файле `/etc/pam.conf`, который в настоящее время считается устаревшим. Во втором поле указан тип модуля: `auth`, `account`, `password` или `session`. Значение третьего поля может быть следующим:

- `required`. Этот флаг указывает на то, что модуль должен завершиться успешно. В противном случае выдается сообщение об ошибке аутентификации, но остальные модули аналогичного типа все равно выполняются, чтобы пользователь не смог узнать, в каком именно модуле возникла ошибка.
- `requisite`. Этот флаг указывает на то, что модуль должен завершиться успешно. В противном случае выдается сообщение об ошибке аутентификации и выполнение других модулей аналогичного типа немедленно прекращается.
- `sufficient`. Этот флаг указывает на то, что в случае успешного завершения модуля все остальные модули аналогичного типа игнорируются.
- `optional`. Этот флаг указывает на то, что код завершения модуля игнорируется. Только если это единственный модуль в стеке или другие модули аналогичного типа возвращают код `PAM_IGNORE`, код завершения модуля возвращается приложению.

Чаще всего используется флаг `required`, поскольку в этом случае обеспечивается наиболее безопасная конфигурация. Злоумышленник не сможет опробовать различные варианты регистрации в системе, чтобы понять, в чем причина ошибки.

В поле *модуль* задается имя модуля PAM (при необходимости — еще и путь к нему). Некоторые модули принимают дополнительные параметры, которые можно указывать в последнем поле. Приведем пример конфигурационного файла:

```
auth        required    pam_nologin.so  no_warn
auth        required    pam_unix.so     no_warn try_first_pass
account     required    pam_unix.so
session     required    pam_unix.so
password    required    pam_unix.so     no_warn try_first_pass
```

На заметку

Таков формат файла в каталоге /etc/pam.d. В файле /etc/pam.conf в начале каждой строки будет указано название службы, например login. В конфигурационных файлах могут также присутствовать строки комментариев, начинающиеся с символа '#'. Часто в них задаются альтернативные варианты конфигурации, которые можно активизировать, удалив знак комментария.

В этом примере два модуля PAM (pam_nologin.so и pam_unix.so) используются для аутентификации, и один модуль (pam_unix.so) — для управления учетными записями, сеансами и паролями. На практике модуль pam_unix.so является основным компонентом большинства конфигураций. Разрешается подключать дополнительные компоненты, которые не входят в стандартную конфигурацию. Например, можно добавить поддержку аутентификации средствами Kerberos.

На заметку

Kerberos — это система аутентификации, используемая в крупных сетях. Она допускает централизованное управление базами паролей и предоставляет услуги шифрования. Дополнительную информацию можно найти по адресу http://web.mit.edu/kerberos/www/.

Настройка стандартных параметров регистрации

Как указывалось выше, при регистрации в текстовом режиме выполняются специальные сценарии, а также считываются специальные конфигурационные файлы, устанавливающие переменные среды пользователя. В этом процессе участвуют файлы двух уровней. Первые хранят общесистемные установки по умолчанию, вторые относятся к конкретным пользователям. Мы в первую очередь поговорим о глобальных конфигурационных файлах.

В текстовом режиме ключевую роль играет конфигурационный файл интерпретатора команд пользователя. Ниже перечислены основные интерпретаторы и их конфигурационные файлы.

- **sh**. Стандартный интерпретатор команд FreeBSD, который используется, если не указано иное. Его глобальный конфигурационный файл называется /etc/profile.
- **tcsh**. Интерпретатор tcsh довольно популярен. Его глобальные конфигурационные файлы называются /etc/csh.cshrc и /etc/csh.login. При выходе пользователя из системы интерпретатор выполняет сценарий /etc/csh.logout.
- **bash**. Этот интерпретатор популярен среди пользователей Linux; во FreeBSD он не инсталлируется по умолчанию. Если его установить, он будет работать с тем же файлом /etc/profile, что и интерпретатор sh.

Изначально конфигурационные файлы всех интерпретаторов пусты. Все они являются сценариями, которые можно запрограммировать на выполнение требуемых действий, как описывается в главе 31. Самая распространенная модификация — установка *переменных среды*. Это специальные именованные параметры, доступные для всех программ. Например, интерпретатор команд, проверяя значение переменной PATH, определяет, в каких каталогах следует искать вызываемые программы. Многие программы с помощью переменных среды находят свои конфигурационные файлы и определяют установки по умолчанию. В частности, некоторые программы используют переменную EDITOR, чтобы выяснить, какой редактор нужно запускать для редактирования файла.

В интерпретаторах sh и bash переменные среды задаются следующим образом:

```
export ПЕРЕМЕННАЯ=значение
```

В качестве значения можно указать имя другой переменной среды, поставив перед ним знак '$'. К примеру, в следующей строке к содержимому переменной PATH добавляется каталог /opt/bin:

```
export PATH=$PATH:/opt/bin
```

В интерпретаторе tcsh то же самое делается с помощью команды setenv, а не export, при этом знак равенства опускается. Если делается ссылка на другую переменную, ее имя нужно брать в кавычки. Вот эквивалентный предыдущему пример:

```
setenv PATH "$PATH":/opt/bin
```

Еще одна полезная команда — alias, она назначает заданной команде альтернативное имя. Предположим, в системе есть пользователи, которые привыкли работать в командной строке DOS или Windows. Им привычнее, когда список содержимого каталога отображается с помощью команды dir, поэтому в файл /etc/profile можно добавить такую строку:

```
alias dir='ls -l
```

В случае интерпретатора tcsh команда будет выглядеть так же, только вместо знака равенства будет стоять пробел. Теперь любой пользователь, который введет dir получит расширенный список содержимого каталога, как если бы была введена команда ls -l.

Старайтесь вносить минимум изменений в глобальные конфигурационные файлы. Такие изменения могут сбить с толку пользователей, привыкших к стандартной конфигурации системы. Те же самые модификации лучше делать в конфигурационных файлах отдельных пользователей. Пользователи могут устанавливать требуемые переменные среды и создавать псевдонимы команд, которые считают нужными, не мешая работе других пользователей.

Пользовательские конфигурационные файлы

Пользовательские конфигурационные файлы аналогичны общесистемным и могут содержать те же установки. Дополнительно существуют файлы, позволяющие задавать параметры регистрации в графическом режиме.

Сценарии регистрации в текстовом режиме

После выполнения глобальных конфигурационных сценариев интерпретатор команд запускает пользовательские сценарии конфигурации. Вот они:

- **sh**. Пользовательский конфигурационный файл для интерпретатора sh называется .profile. В большинстве случаев вызывается также сценарий .shrc из начального каталога пользователя. Сценарий .profile выполняется только при входе пользователя в систему, а сценарий .shrc — при каждом запуске интерпретатора (например, когда в графическом режиме открывается терминальное окно).
- **tcsh**. Интерпретатор tcsh последовательно читает файлы .tcshrc, .cshrc, .history, .login и .cshdirs. Последние три сценария выполняются только при входе пользователя в систему.
- **bash**. В процессе регистрации пользователя интерпретатор bash последовательно считывает файлы .bash_profile, .bash_login и .profile. При выходе пользователя из системы выполняется сценарий .bash_logout. Если интерпретатор вызывается не в ходе регистрации, выполнятся сценарий .bashrc.

Эти файлы можно модифицировать так же, как их общесистемные аналоги — добавлять псевдонимы устанавливать переменные среды и т.д. Но поскольку файлы принадлежат конкретным пользователям, их установки не пересекаются и не перекрывают друг друга. Для каждого файла должны быть установлены все биты выполнения. Это делается командой `chmod a+x` *сценарий*.

Для некоторых интерпретаторов команд образцы стандартных конфигурационных файлов хранятся в каталоге `/usr/share/skel`. Имена файлов начинаются не с точки, а со слова `dot`, например `dot.cshrc` При добавлении в систему новой учетной записи эти файлы можно скопировать в начальный каталог пользователя, переименовав должным образом

Сценарии регистрации в графическом режиме

Когда пользователь регистрируется в графическом режиме, система тоже считывает и обрабатывает ряд конфигурационных файлов. Обычно используется один из двух файлов начального каталога пользователя

- **`xinitrc`**. Сценарий инициализации графической среды вызываемый когда пользователь вводит **`startx`** в текстовом режиме
- **`xsession`**. Эквивалент предыдущего сценария, используемый программой XDM

На заметку Если вместо XDM используется программа KDM или GDM то запуск сценария `xsession` зависит от типа сеанса Эти программы позволяют пройти традиционную регистрацию в системе в стиле XDM или запустить одну из нескольких графических оболочек

Как и в случае сценариев текстовой регистрации сценарии графической регистрации должны быть исполняемыми, поэтому выполните в начальном каталоге команду **`chmod a+x .xinitrc`** или **`chmod a+x xsession`**. Эти сценарии тоже могут устанавливать переменные среды и запускать различные программы. Чаще всего вызывается диспетчер окон или графическая оболочка а также ряд вспомогательных программ К примеру, в листинге 6 3 показан сценарий запускающий диспетчер окон IceWM и открывающий терминальное окно

Листинг 6 3 Пример сценария `xinitrc` или `xsession`

```
#!/bin/sh
/usr/X11R6/bin/xterm &
/usr/X11R6/bin/icewm
```

Важно, чтобы программы, вызываемые из сценариев, выполнялись в фоновом режиме Для этого в конце строки запуска программы следует указать оператор `&` В приведенном примере программа `xterm` запускается в фоновом режиме, после чего выполнение сценария продолжается и вызывается программа IceWM. Но в последней строке оператор `&` отсутствует Это означает, что выполнение сценария прекращается до тех пор, пока программа IceWM не завершится. Такая конфигурация является типичной и имеет два важных последствия

- *Ни одна программа, вызываемая в сценарии после IceWM не будет выполнена до тех пор, пока не завершит работу программа IceWM* Это позволяет включать в сценарий вызовы программ очистки выдавать звуковой сигнал при выходе из системы и т.д.
- *Программа IceWM управляет X сеансом* Когда программа IceWM завершает работу, выполнение сценария продолжается После вызова диспетчера окон обычно выполняется очень небольшое число команд Когда завершается сам

сценарий, X-сеанс заканчивается и пользователь возвращается к экрану регистрации, отображаемому XDMCP-программой

Выполняющиеся относительно долго программы должны работать в фоновом режиме, чтобы не задерживать выполнение остальной части сценария Обычно признаком конца графического сеанса служит завершение работы диспетчера окон, но это может быть и другая программа Например, если в листинге 6 3 поменять местами вызовы `xterm` и `icewm` и запустить первую программу в интерактивном режиме а вторую — в фоновом, то X-сеанс будет завершаться при закрытии терминального окна.

Если сеанс начинается командой `startx` а в системе нет файла `.xinitrc`, проверяется файл `/etc/X11/xinit/xinitrc`. Если же программа XDM не находит файл `.xsession` выполняется сценарий `/usr/X11R6/lib/X11/xdm/Xsession`. Он запускает диспетчер окон `twm` и программу `xsm` в качестве диспетчера X-сеанса, когда в меню программы выбирается команда Shutdown, сеанс завершается. Стандартные файлы `.xinitrc` и `.xsession` существуют также в каталоге `/usr/share/skel`. Их можно копировать в пользовательские каталоги при создании новых учетных записей Оба сценария запускают оболочку GNOME

Запуск программ в запланированное время: демон cron

Бывают ситуации, когда нужно выполнять программы в четко определенное время в отсутствие человека. Автоматический запуск таких программ поручается системе Для этого во FreeBSD предусмотрено два средства: `at` и `cron`. Команда `at` выполняет одноразовые действия, например запускает ресурсоемкую программу поздно ночью Демон `cron` более сложен. Он способен многократно выполнять одни и те же задания, например загружать почту с сервера каждый час. Система изначально сконфигурирована таким образом, что ряд заданий, связанных с повседневным регламентным обслуживанием, контролируется демоном `cron` без вмешательства пользователя

Команда at

Команда `at` позволяет выполнить заданную программу один раз в требуемое время. Существует схожая команда `batch`, заставляющая систему выполнить программу когда показатель средней загруженности упадет ниже определенного уровня (в частности, ниже 1,5) Синтаксис команд таков.

```
at [-V] [-q очередь] [-f файл] [-mldbv] время
batch [-V] [-q очередь] [-f файл] [-mv] [время]
```

Ниже описаны аргументы команд

- `-V` Отображение номера версии программы
- `-q` *очередь* Поддерживается множество очередей заданий, каждая из которых идентифицируется буквой (от `a` до `z` и от `A` до `Z`) Обычно очереди команды `at` помечаются строчными буквами (по умолчанию — `c`), а очереди команды `batch` — прописными (по умолчанию — `E`) Чем дальше буква от начала алфавита, тем меньше приоритет очереди.
- `-f` *файл*. Как правило, планируемую команду указывают непосредственно в командной строке, после чего нажимают <Ctrl+D> (признак конца файла). Но можно задать команду в файле, передав его имя в качестве аргумента Это особенно удобно в случае сценариев

- **-m**. Если эта опция присутствует, то по завершении задания пользователю за пустившему его, посылается почтовое сообщение.
- **-l**. Эта опция заставляет команду `at` вывести список заданий, находящихся в очереди. Существует эквивалентная команда `atq`.
- **-d**. Эта опция вызывает удаление задания из очереди. То же самое делает команда `atrm`.
- **-b**. Эта опция заставляет команду `at` работать в стиле `batch`.
- **-v**. Если присутствует опция `-l`, то опция `-v` заставляет команду вывести список заданий, уже завершившихся, но еще не удаленных из очереди.
- *время*. Плановое время выполнения задания. Поддерживается несколько форматов времени, включая *ЧЧмм* и *ЧЧ:мм* с необязательным суффиксом `AM` или `PM` для выполнения задания в ближайшие 24 часа. Распознаются также ключевые слова `noon` (полдень), `midnight` (полночь) и `teatime` (16.00). Если задание планируется на более поздний срок, нужно дополнительно указать дату. Поддерживаются спецификации *ММддгг*, *ММ.дд.гг*, *ММ/дд/гг* и *название_месяца день год*. Можно задавать время в относительном формате, указывая знак '+', числовое значение и единицу измерения, например `+30minutes`.

По умолчанию команды `at` и `batch` доступны только суперпользователю. Для отмены этого ограничения нужно создать файл `/var/at/at.allow` или `/var/at/at.deny`. В первом файле указываются имена пользователей, которым тоже разрешено планировать задания. Если существует файл `at.deny`, то планирование заданий разрешено всем пользователям, кроме тех, чьи имена присутствуют в этом файле.

Рассмотрим пример команды `at`:

```
$ at -m +2 minutes
/bin/ls > ~/listing.txt
```

По окончании ввода команды нужно нажать <Enter> или <Ctrl+D>. Через две минуты в вашем начальном каталоге появится файл `listing.txt`, содержащий результаты работы команды `ls`. Кроме того, будет получено почтовое сообщение о том, что выполнение команды завершено.

На заметку Для обработки очереди команды `at` создается задание демона `cron`. По умолчанию это задание выполняется каждые 5 минут, так что в рассмотренном примере на выполнение команды может уйти больше двух минут.

Функции демона cron

Часто одно и то же задание должно регулярно выполняться в определенное время. Например, можно с помощью программы `fetchmail` (описана в главе 19) проверять почту на сервере каждый час или использовать сервер точного времени (см. главу 22) для синхронизации системных часов раз в день. В этих ситуациях на помощь приходит демон `cron`. Он заставляет систему выполнять требуемые задания через постоянные промежутки времени.

Демон `cron` обладает достаточно гибкими средствами планирования. Можно запускать задания хоть каждую минуту, пусть на практике это применяется редко. Задания могут выполняться с разной периодичностью в зависимости от дня недели или времени суток.

Существуют два базовых типа планируемых заданий: *системные* (их может создавать только суперпользователь) и *пользовательские* (создаются рядовыми пользовате

лями). Их функциональные возможности идентичны Системные задания создаются путем редактирования файлов в каталоге /etc, а для создания и редактирования пользовательских заданий предназначена утилита crontab

На заметку Во FreeBSD есть стандартный набор системных заданий, выполняющих, к примеру, ротацию журнальных файлов, чтобы они не переполнили диск Задания, для выполнения которых не требуются привилегии суперпользователя или другие административные полномочия вводятся рядовыми пользователями

Демон cron — один из многих, вызываемых из сценариев запуска системы (в частности, из сценария /etc/rc). Он работает постоянно, проверяя очередь каждую минуту чтобы узнать, нет ли заданий, которые нужно выполнить Если таковые отсутствуют демон делает минутную паузу

Планирование системных заданий

Основной управляющий файл демона cron называется /etc/crontab Он начинается с нескольких операций присваивания переменных, например:

```
PATH=/etc:/bin./sbin:/usr/bin /usr/sbin
```

В этих строках формируется среда выполнения демона cron, чтобы не нужно было указывать полный путь к выполняемым программам и чтобы программы могли работать с привычными переменными среды Строки комментариев начинаются с символа '#' и игнорируются демоном

Основные записи файла имеют следующий синтаксис

```
минута час день_месяца месяц день_недели пользователь команда
```

Первые 5 полей задают время выполнения команды Поддерживается несколько форматов

- **Непрерывное выполнение** Звездочка ('*) соответствует любому значению данного поля Например, если задание должно выполняться каждый день укажите в качестве дня недели и дня месяца символ '*

На заметку Если конкретно указан день месяца и день недели демон cron выполняет команду, когда *любое* из полей соответствует текущему дню. Если же указано одно из значений, то команда выполняется только в нужный день недели или месяца несмотря на то что в другом поле задано непрерывное выполнение.

- **Целые числа** Целое число например 7 или 23, трактуется буквально. В поле *день_недели* значения 0 и 7 соответствуют воскресенью Значение поля *час* задается в 24-часовом формате.
- **Символические обозначения** В полях *месяц* и *день_недели* можно указывать названия, а не числа. Используйте первые три буквы названия месяца или дня недели, например Nov (ноябрь) или Tue (вторник). Регистр символов не важен
- **Диапазоны** В любом из первых пяти полей можно задать диапазон значений с помощью дефиса, например Mon-Fri в поле *день_недели* или 8-17 в поле *час*
- **Списки** Списки значений разделяются запятыми Например, если в поле *минута* стоит выражение 0,20,40, задание будет выполняться каждые 20 минут
- **Пошаговое приращение.** После звездочки или спецификации диапазона можно через косую черту указать шаг приращения Например, если в поле *час* стоит выражение */3, задание будет выполняться каждые три часа.

В поле *пользователь* указывается пользователь от имени которого выполняется за дание Для системных заданий здесь обычно стоит имя root. Наконец в поле *команда* указывается выполняемая команда или сценарий Рассмотрим следующую типичную запись.

```
04,34 8 17 * * mon-fri mailnull fetchmail
```

Она заставляет демон cron выполнять команду fetchmail дважды в час (в 4 и 34 минуты), с 8:04 до 17:34, с понедельника по пятницу, от имени пользователя mailnull (это стандартная низкоприоритетная учетная запись для операций с элек тронной почтой)

В этом примере демонстрируется довольно разумный подход к планированию. Пользователи часто планируют выполнение заданий на начало или середину часа, не учитывая, что другие пользователи могут делать то же самое. Одновременный запуск заданий множества пользова телей перегружает сервер Этого легко избежать, если смещать время на несколько минут относительно начала или середины часа. Конечно, это не актуально в случае чисто локальных заданий, но удобно для таких заданий, как загрузка почты или синхронизация часов с внешним сервером точного времени

Многие задания обрабатываются с помощью следующих стандартных строк в фай ле /etc/crontab

```
# ежедневное/еженедельное/ежемесячное обслуживание
1       3       *       *       *       root    periodic daily
15      4       *       *       6       root    periodic weekly
30      5       1       *       *       root    periodic monthly
```

Здесь вызывается утилита periodic, которая выполняет сценарии, хранящиеся в подкаталогах каталога /etc/periodic, в частности /etc/periodic/daily, /etc/periodic/weekly и /etc/periodic/monthly. Таким образом, легко создать системное задание, добавив сценарий, выполняющий нужное действие, в соответст вующий каталог. Во FreeBSD эти сценарии называются *очередность.имя* где *очередность* — это трехзначный порядковый номер, а *имя* — это название задания Задания выполняются в соответствии с заданной очередностью

Планирование пользовательских заданий

Рядовые пользователи могут планировать выполнение заданий с помощью коман ды crontab, синтаксис которой таков:

```
crontab [-u пользователь] [-l | -r    e] [файл]
```

Обычно выполняется команда **crontab** *файл*, где в качестве аргумента задается имя инсталлируемого файла Его формат практически идентичен формату системного файла /etc/crontab. Разница лишь в том, что имя пользователя не указывается. Если файл с таким именем уже существует, он заменяется обновлением вариантом.

Опции -l r и -e заставляют команду, соответственно, отобразить текущий файл удалить файл и открыть его для редактирования. В последнем случае вызывается ре дактор заданный в переменной среды EDITOR (по умолчанию это vi) Если присутст вует одна из этих опций, имя файла указывать не нужно: система работает с файлом который уже инсталлирован соответствующим пользователем.

В распоряжении суперпользователя есть опция -u *пользователь*, позволяющая заменять, просматривать, удалять и редактировать пользовательские crontab-файлы Рядовым пользователям эта опция недоступна Одно из назначений опции — плани рование заданий от имени системных учетных записей, таких как daemon или

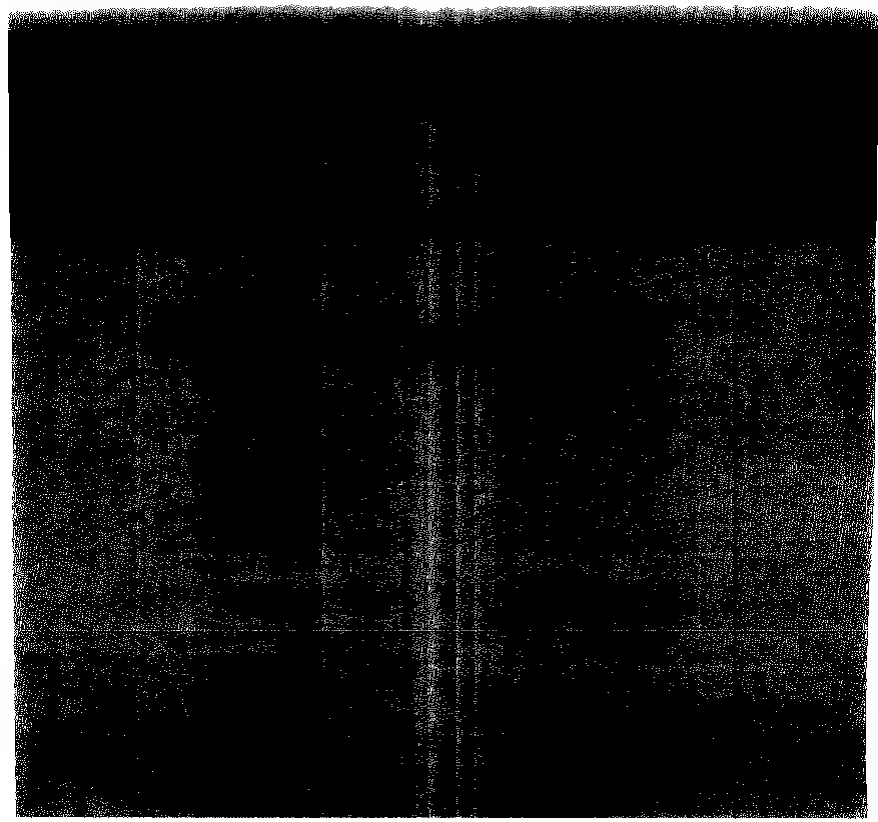
mailnull. Эти учетные записи обычно не используются для регистрации в системе поэтому только суперпользователь может создавать для них задания. Ну и естественно, опция -u важна для контроля пользовательских crontab-файлов

Резюме

Одна из основных функций FreeBSD — выполнение программ Обычно пользователь вводит имя программы в командной строке или щелкает на значке программы в графической среде. Но некоторые важные программы запускаются по-другому В частности само ядро FreeBSD запускается загрузчиком Оно вызывает демон init, который выполняет сценарии запуска системы. Из этих сценариев запускаются много численные демоны и конфигурационные программы, определяющие, как система будет работать непосредственно после загрузки Важно уметь настраивать этот процесс чтобы контролировать работу системы

При регистрации пользователя в системе тоже выполняются различные программы Какие именно — зависит от способа регистрации в системе Эти программы формируют среду, в которой работает пользователь Наконец, во FreeBSD можно планировать запуск программ в заданное время Это может быть одноразовое (команды at и batch) либо периодическое (демон cron) выполнение. Утилиты планирования позволяют эффективнее распределять нагрузку на систему и автоматизировать важные системные операции

Полный справочник по



В главе 2 вкратце рассказывалось о том как во FreeBSD выполняется разбивка на разделы. В этой главе тема дисковых разделов рассматривается более подробно Мы поговорим о том, зачем разбивать диск на разделы, какие существуют средства для создания разделов и манипулирования ими и каковы базовые принципы организации каталогов FreeBSD.

Разделы: хранилища файлов данных

Дисковый раздел напоминает ящик в картотечном шкафу: в нем может храниться ограниченное число файлов и папок (каталогов). Если ящик или раздел переполняется нужно переместить часть файлов в другой ящик или раздел Размер раздела может быть самым разным от сотен килобайт до нескольких гигабайт.

Зачем разбивать диск на разделы

Как описывалось в главе 2, жесткие диски разбивают на разделы, чтобы предоставить различным операционным системам отдельные области для хранения данных создать в рамках одной операционной системы несколько областей хранения с разными характеристиками, изолировать данные во избежание ил повреждения неуправляемыми процессами и т д Схема разбивки на разделы выбирается при инсталляции FreeBSD и после этого обычно не меняется. Но бывают ситуации, когда разбивка на разделы необходима

- **Добавление диска.** Если текущий диск переполняется, проходится покупать дополнительный. Новый диск нужно разбить на разделы, добавив их в существующее дерево каталогов
- **Замена диска.** Если текущий диск имеет слишком маленькую емкость и не удовлетворяет потребностям системы, его приходится менять Новый диск нужно разбить на разделы и скопировать на него старую систему, временно подключив старый диск Это можно сделать с помощью команды `tar`, которая перенесет данные со старого диска на новый. Затем необходимо воспользоваться загрузочной дискетой для инсталляции системного загрузчика на новый диск К примеру, если новый диск смонтирован в каталоге `/mnt`, то следующая команда копирует на него систему хранящуюся в разделах / (корневой), `/usr` и `/home`

```
# tar cplf   / /usr/ /home | (cd /mnt; tar xvpf  )
```

- **Перераспределение разделов.** Самая сложная заяача — это перераспределение существующих разделов, когда обнаруживается, что первоначальные предположения о размерах разделов оказадись неверны Существуют коммерческие и свободно распространяемые утилиты, позволяющие решить задачу, не разрушая данные файловых систем, но в случае файловой системы FFS возможности этих утилит ограничены. Стандартная утилита `growfs` может увеличить размер пространства занимаемого FFS Она требует, чтобы предварительно был изменен размер раздела с помощью команды `disklabel` и чтобы после этого раздела на диске было свободное пространство Из-за подобного ограничения часто приходится искать временное место для хранениил данных Это может быть другой жесткий диск, съемный диск, файл-сервер или магнитная лента После повторной разбивки диска на разделы данные приходится копировать обратно

- **Использование съемных носителей.** Некоторые съемные носители, например дискеты не разбиваются на разделы. Другие, как, скажем, Zip-диски разбиваются, причем часто они уже продаются с разбивкой на разделы которую может понадобиться изменить особенно если для хранения данных планируется использовать FFS.

Разбивка на разделы, как правило, связана с уничтожением данных если только не применяется утилита неразрушающей разбивки, такая как PartitionMagic (`www.powerquest.com`), GNU Parted (`http://www.gnu.org/software/parted/`) или `growfs`. Таким образом, схему разбивки на разделы лучше выбирать один раз и не менять долгое время

Когда выполняется разбивка на разделы или создается файловая система, будьте очень осторожны при выборе файла устройства Очень легко ошибиться, и все данные раздела или даже целого диска будут уничтожены

Создание разделов

Существует множество утилит для создания разделов Некоторые из них уже рассматривались в главе 2. Тем не менее будет полезно вернуться к этой теме и подробнее рассказать о применении различных утилит

Использование утилит сторонних производителей

Во многих операционных системах есть программа `fdisk` или `FDISK`, предназначена для разбивки диска на разделы Во FreeBSD тоже есть утилита `fdisk`, описываемая далее Утилиты сторонних производителей полезны для подготовки диска к инсталляции нескольких операционных систем или в тех случаях, когда нужно на съемный диск установить не FFS, а другую файловую систему Такие утилиты обычно проще, чем `fdisk`, хотя интерфейс программы `sysinstall` и упрощает ее применение

Существуют также программы разбивки на разделы, не входящие в состав той или иной операционной системы. Среди них можно назвать PartitionMagic и GNU Parted Такие программы обладают более широкими возможностями, чем стандартные утилиты операционных систем, и, в частности поддерживают неразрушающую разбивку на разделы.

С помощью утилит сторонних производителей удобно создавать разделы x86, но не BSD-разделы Если создаваемый раздел предназначен для FreeBSD, нужно сделать его первичным и присвоить ему код 165 (0xA5) FreeBSD распознает такой раздел как свой первичный, после чего в нем можно будет создавать BSD-разделы Можно просто оставить на диске пустое место и воспользоваться утилитами FreeBSD для создания первичного раздаяа и подразделов

Использование программы sysInstall

В главе 2 описывалось, как с помощью программы `sysinstall` создать первичный раздел и подразделы на этапе инсталляции системы Эту программу проще всего использовать для разбивки на разделы второго жесткого диска или съемного носителя.

После инсталляции FreeBSD средства программы `sysinstall` предназначенные для разбивки на разделы, вызываются следующим образом

1 Перейдите в режим суперпользователя и введите **`/stand/sysinstall`**. В результате будет запущена программа `sysinstall` и появится ее главное меню (рис 7 1)

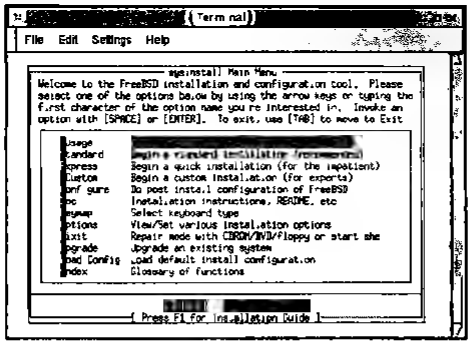


Рис 7 1 Главное меню программы sysinstall

2 Выберите пункт `Configure` и нажмите <Enter> Появится меню конфигурирования FreeBSD (рис 7 2)

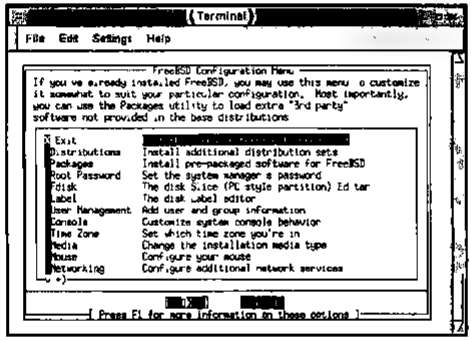


Рис 7 2 Меню конфигурирования системы

3 Команда `Fdisk` предназначена для редактирования таблицы разделов x86 После ее выполнения будет автоматически активизирована следующая команда

4 Команда `Label` предназначена для редактирования таблицы разделов FreeBSD (так называемая *дисковая метка*) в пределах первичного раздела Это позволяет создать подразделы FreeBSD

Два последних этапа описывались в главе 2 По завершении работы с программой FreeBSD Disklabel Editor нажмите **w** чтобы сохранить сделанные изменения Далее будет запущена утилита `newfs`, которая создаст файловые системы в новых разделах Таким образом, под контролем программы `sysinstall` находятся все этапы разбивки на разделы создание первичного раздела FreeBSD, создание подразделов в первичном разделе и создание файловых систем на этих разделах.

Использование утилит командной строки

Программа `sysinstall` — удобное средство создания разделов но иногда приходится изучать используемые ею низкоуровневые команды Если речь идет о разделах, то таковых три: `fdisk`, `disklabel` и `newfs` Они управляют созданием таблицы первичного раздела x86 метки BSD-диска в первичном разделе FreeBSD и файловых систем в BSD-разделах.

Утилита fdisk

Как и одноименные программы других операционных систем, утилита `fdisk` позволяет просматривать и редактировать таблицу разделов x86. Она может выводить сведения о таблице разделов, создавать новую таблицу на основании данных из конфигурационного файла и записывать данные в строго определенном порядке на основании информации, полученной от пользователя Если нужно поменять разделы в произвольном порядке, воспользуйтесь программой `sysinstall` или программой разбивки на разделы стороннего производителя

Утилита `fdisk` вызывается одним из двух способов

```
fdisk  f конфигурационный_файл [-itv] [диск]
fdisk [-BIaistu] [ b код_загрузки] [-1234] [диск]
```

В первом варианте утилита собирает данные из конфигурационного файла, хранящегося на диске Если вам регулярно приходится разбивать на разделы диски примерно одинаковой емкости создайте файл с описанием схемы разбивки этих дисков. Кроме того что это упростит работу, вы будете делать меньше ошибок Формат файла описан на man-странице утилиты `fdisk`

Во втором варианте утилита либо отображает информацию о диске, либо позволяет отредактировать таблицу разделов. Если задать только целевой диск (например `/dev/ad0` для первого диска EIDE или `/dev/da1` для второго SCSI-диска), команда выведет данные о таблице разделов

```
# fdisk /dev/ad0
****************** Working on device /dev/ad0 ***********
parameters extracted from in-core disklabel are:
cylinders=784 heads=255 sectors/track=63 (16065 blks/cyl)

Media sector size is 512
Warning: BIOS sector numbering starts with sector 1
Information from DOS bootblock is:
The data for partition 1 is:
sysid 12,(DOS or Windows 95 with 32 bit FAT, LBA)
    start 63, size 4192902 (2047 Meg), flag 0
       beg: cyl 0/ head 1/ sector 1;
       end: cyl 260/ head 254/ sector 63
The data for partition 2 is:
sysid 165,(FreeBSD/NetBSD/386BSD)
    start 4192965, size 8401995 (4102 Meg), flag 80 (active
       beg: cyl 261/ head 0/ sector 1;
       end: cyl 783/ head 254/ sector 63
The data for partition 3 is:
<UNUSED>
The data for partition 4 is:
<UNUSED>
```

В отчете утилиты приведены сведения о геометрии диска (распределение цилиндров, головок и секторов), о размере сектора и о четырех первичных разделах В частности раздел 1 — это FAT-раздел DOS или Windows размером 2047 Мбайт, раздел 2 – это первичный раздел FreeBSD размером 4102 Мбайт Разделы 3 и 4 не определены

На заметку Утилита `fdisk` во FreeBSD не выдает информацию о логических разделах и не позволяет работать с ними

Ниже описаны опции утилиты `fdisk`

- `B` — повторная запись загрузчика в ГЗЗ. Это помогает устранять ряд проблем
- `b` *`код_загрузки`* — запись кода загрузчика в ГЗЗ из заданного файла (по умолчанию `/boot/mbr`).
- `-u` — обновление ГЗЗ. Утилита спрашивает у пользователя, какие изменения нужно сделать, позволяя редактировать информацию о разделах и геометрии диска
- `-I` — создание таблицы разделов, содержащей только первичный раздел FreeBSD, охватывающий весь диск. Это самый быстрый способ подготовки диска к установке FreeBSD
- `-i` — инициализация ГЗЗ. Утилита запрашивает у пользователя необходимую информацию, как и в случае опции `-u`
- `f` *`конфигурационный_файл`* — чтение информации о разделах из конфигурационного файла
- `-a` — изменение только активного раздела
- `s` — вывод информационной сводки и завершение работы
- `-t` — тестовый режим, выполняются все традиционные действия, но не происходит запись данных
- `v` — в сочетании с опцией `f` заставляет утилиту отобразить таблицу разделов
- от `-1` до `-4` — задают раздел, с которым ведется работа

На практике большинство изменений делается с помощью команды `fdisk -i /dev/`*`идентификатор_диска`* или `fdisk -u /dev/`*`идентификатор_диска`*. Эти команды выводят примерно такой отчет, как показано выше, но при этом утилита запрашивает, нужно ли изменить тот или иной параметр. В конце утилита предлагает сохранить изменения, чтобы в случае ошибки пользователь мог отказаться от сделанных установок. Поэкспериментируйте с утилитой в тестовом режиме, чтобы лучше узнать ее возможности.

Утилита disklabel

Утилита `disklabel` позволяет просмотреть и отредактировать таблицу, хранящуюся в первичном разделе FreeBSD, — дисковую метку. Утилиту можно вызывать по-разному, добиваясь разных результатов. В упрощенном виде синтаксис вызова выглядит так:

```
disklabel [ -wenWN] диск [тип_диска | auto]
```

Рассмотрим опции утилиты

- `w` — запись новой метки на диск. Эта метка не содержит записи о разделах. Обычно нужно задать тип диска или указать ключевое слово `auto`.
- `e` — модификация существующей дисковой метки в редакторе, который задан в переменной `EDITOR`. При выходе из редактора сделанные изменения будут сохранены
- `n` — вывод информационной сводки без записи изменений на диск
- `W` — разрешает запись в область диска, где хранится метка. FreeBSD часто запрещает запись метки, поэтому приходится использовать эту опцию.

- **-N** — запрещает запись в область диска, занимаемую меткой. Эта опция отменяет опцию **-W**.
- *диск* — задает модифицируемый диск. Обычно это идентификатор раздела *x86*, например **/dev/ad0s2** или **/dev/da0s1**. Компонент **/dev/** можно опустить, иногда это даже необходимо, например, когда задается тип **auto**.
- *тип диска* | **auto** — файл **/etc/disktab** хранит набор установок для наиболее распространенных типов дисков. Можно выбрать требуемый тип диска или указать ключевое слово **auto**, чтобы система автоматически сгенерировала информацию о диске. В файле **disktab** описаны не только обычные жесткие диски, но и популярные съемные носители, например **zip100** и **orb2200**.

Как правило, при подготовке нового диска к использованию нужно выполнить как минимум две команды **disklabel**: для создания новой таблицы разделов FreeBSD и для ее редактирования. Дополнительно могут понадобиться еще две — начальная и завершающая — команды с опциями **-W** и **-N**, если окажется, что изменения не вступают в силу. Показанная ниже последовательность команд подключает Zip-диск к устройству **/dev/da0s4**.

```
# disklabel -w da0s4 auto
# disklabel -e /dev/da0s4
```

Эти команды просты лишь на первый взгляд, ведь во втором случае нужно вручную отредактировать информацию, хранящуюся в дисковой метке. Файл, загружаемый в редактор, начинается со строк, в которых приводятся данные о геометрии диска и количестве секторов. Ближе к концу файла находятся определения разделов. В новой системе будет присутствовать всего один раздел с типом файловой системы (поле **fstype**) **unused**. Более сложная разбивка выглядит так:

```
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  a:   262144        0    4.2BSD     1024  8192    22   # (Cyl.    0 - 16*)
  b:   213163   262144      swap                         # (Cyl.  16*- 29*)
  c:  8401995        0    unused        0     0         # (Cyl.    0 - 522)
  e:   307200   475307    4.2BSD     1024  8192    23   # (Cyl.  29*- 48*)
  f:   307200   782507    4.2BSD     1024  8192    23   # (Cyl.  48*- 67*)
  g:  3407872  1089707    4.2BSD     1024  8192    22   # (Cyl.  67*- 279*)
  h:  3904416  4497579    4.2BSD     1024  8192    22   # (Cyl. 279*- 522*)
```

В первой колонке указан идентификатор раздела, во второй — размер раздела в блоках (обычно блок — это 512 байт, поищите в начале файла строку, помеченную как **bytes/sector**). В поле **offset** задается точка начала раздела, а в поле **fstype** — тип файловой системы. Для разделов FFS в четвертом поле должно стоять **4.2BSD**. Колонки **fsize**, **bsize** и **bps/cpg** задают системно зависимые значения. Старайтесь использовать значения по умолчанию (значение в последней колонке не играет особой роли). В конце строки после знака '**#**' стоят комментарии. В них указано, в каких цилиндрах расположен раздел. Эту информацию можно проигнорировать.

Идентификаторы разделов имеют конкретное значение. Идентификатор **a** зарезервирован за корневым разделом, **b** — за областью подкачки, **c** ссылается на весь диск. Идентификатор **d** сейчас не используется, а раньше он тоже был зарезервирован, и некоторые утилиты плохо работают с разделами, помеченными как **d**. Остальные буквенные идентификаторы доступны для обычных файловых систем.

Определяя разделы таким способом, нужно быть очень осторожным, чтобы не создать перекрывающиеся разделы. Тщательно проверяйте значения, указанные в колонках **size** и **offset**. Смещение нового раздела должно равняться сумме значений **offset** и **size** предыдущего раздела. Исключением является раздел **c**, который фактически не существует, по крайней мере с точки зрения вычисления смещений.

Утилита `disklabel` обладает очень широкими возможностями но и вероятность повреждения данных тоже очень высока Старайтесь не вызывать утилиту непосредственно а использовать в качестве интерфейсной надстройки программу `sysinstall` Необходимость в утилите возникает, только когда приходится вносить необычные изменения в конфигурацию разделов

Утилита newfs

Утилита `newfs` позволяет создать новую файловую систему формата FFS Синтаксис утилиты таков

```
newfs [-NOU] [-S размер_сектора] [-b размер_блока
 [-f размер_фрагмента] [ g размер_файла] [ i размер_дескриптора]
 -m процент] [-o space time] -s размер] [-v] устройство
```

На заметку Здесь представлены не все опции утилиты Остальные опции связаны с оптимизацией, основанной на сведениях о геометрии диска что бессмысленно для современных жестких дисков

Назначение опций утилиты таково

- `N` — заставляет утилиту отобразить параметры файловой системы не создавая ее саму
- `-O` — заставляет утилиту создать файловую систему формата 4.3BSD. Эта опция наиболее часто используется когда система загружается с помощью старой версии BIOS.
- `-U` — разрешает *нестрогие обновления* — стратегию оптимизации которая ускоряет восстановление после сбоя
- `-S` *размер_сектора* — задает размер сектора в байтах Значение по умолчанию — 512, и менять его почти никогда не следует
- `-b` *размер_блока* — задает размер блока файловой системы Это минимальный объем дискового пространства, который FreeBSD может выделить файлу. Размер блока должен представлять собой степень числа 2. Минимально допустимое значение — 4096, значение по умолчанию — 16384. Использование блока большего размера приводит к незначительному повышению производительности при работе с крупными разделами, но при этом расходуется дополнительное дисковое пространство
- `-f` *размер_фрагмента* — задает размер фрагмента в байтах Это значение должно представлять собой степень числа 2 и находиться в интервале от *размер_блока*/8 до *размер_блока*. Значение по умолчанию — 2048
- `-g` *размер_файла* — задает ожидаемый средний размер файла
- `-i` *размер_дескриптора* — задает количество байтов, занимаемых индексным дескриптором У каждого файла свой дескриптор, поэтому данный параметр, по сути, налагает ограничение на число файлов в системе По умолчанию создается один индексный дескриптор на каждые 4×*размер_фрагмента* байтов дискового пространства Обычно этого вполне достаточно
- `-m` *процент* задает процент дискового пространства, зарезервированный для суперпользователя Значение по умолчанию — 8, что дает суперпользователю резервное пространство для восстановления данных. Если не указать эту опцию, то в случае переполнения диска суперпользователь даже не сможет загрузиться в системе для устранения проблемы.

- **-o space|time** — заставляет утилиту оптимизировать файловую систему либо с точки зрения рационального использования дискового пространства (чтобы можно было создать наибольшее количество файлов), либо с точки зрения производительности (чтобы доступ к файлам осуществлялся максимально быстро). По умолчанию, если параметр *процент* больше или равен 8, выполняется оптимизация по производительности, а если меньше 8 — оптимизация по использованию дискового пространства.
- **-s** *размер* — задает размер файловой системы в секторах. Это значение обычно определяется на основании размера раздела.
- **v** — заставляет утилиту создать систему на носителе, где нет разбивки на разделы. Это может быть, к примеру, дискета.
- *устройство* — задает имя устройства, например `/dev/da0s4e` или `/dev/ad1s2a`.

Большинство этих опций в обычных ситуациях не нужно. Утилита `newfs` использует вполне разумные установки по умолчанию. Как правило, достаточно такой команды:

```
# newfs /dev/da0s4e
```

Эта команда создает файловую систему на устройстве `/dev/da0s4e`.

Совет При работе со съемными носителями, например Zip-дисками, обычно задается опция `m 0`. Такая установка увеличивает пространство для хранения данных, что важнее всего для съемных устройств.

Во FreeBSD есть похожая утилита `newfs_msdos`, которая создает файловую систему формата FAT. Наиболее важные ее опции — `-F` *тип* (аргумент *тип* может принимать значения 12, 16 или 32 и определяет битовую разрядность данных, с которыми работает система) и `L` *метка* (задает дисковую метку для раздела FAT). Ни одна из них не является обязательной. Утилита вычисляет размер файловой системы на основании размера раздела или емкости устройства, а метка может быть пустой, хотя иногда она помогает идентифицировать диск в Windows.

Команда `newfs msdos` используется для создания файловых систем формата FAT на дискетах и других съемных носителях. Она может также оказаться полезной для выделения места под Windows на новом диске. В разделах FreeBSD файловые системы формата FAT обычно не создаются.

На заметку Применительно к дискете DOS- и Windows-команда `FORMAT` выполняет как низкоуровневое (подготовка диска к записи данных), так и высокоуровневое (создание файловой системы) форматирование. Во FreeBSD это две разные операции. Если дискета ни разу не форматировалась, то для низкоуровневого форматирования необходимо выполнить команду **`fdformat /dev/fd0`**. Затем с помощью команды `newfs` или `newfs_msdos` можно создать файловую систему формата FFS или FAT. Для жесткого диска низкоуровневое форматирование не требуется, поскольку оно выполняется в заводских условиях.

Монтирование разделов

После того как создан раздел, а в нем — файловая система, нужно смонтировать раздел, чтобы пользователи могли с ним работать. Это можно сделать двумя способами.

- временно смонтировать раздел с помощью команды `mount`;
- сконфигурировать FreeBSD на автоматическое монтирование раздела, создав запись для него в файле `/etc/fstab`.

На заметку

В некоторых ситуациях можно получить доступ к файловой системе не монтируя ее. В частности, утилиты пакетов Mtools (`http://mtools.linux.lu`) и HFS Utilities (`http://www.mars.org/home/rob/proj/hfs/`) предоставляют такой доступ к файловым системам FAT (DOS/Windows) и HFS (Macintosh). Чаще всего они используются для доступа к дискетам, иногда — к Zip-дискам или схожим съемным носителям. Некоторые компакт-диски, предназначенные для Macintosh, содержат файловую систему формата HFS, поэтому пакет HFS Utilities может оказаться полезным для чтения файлов с таких компакт дисков.

Команда mount

Если нужно временно смонтировать раздел, например в целях тестирования или для доступа к съемному носителю, воспользуйтесь командой `mount`. Она подключает файловую систему к *точке монтирования*, которая представляет собой обычный каталог. Через этот каталог содержимое файловой системы станет доступно пользователям. Команда имеет следующий синтаксис:

```
mount [-adfpruvw] [-o параметры] [-t тип] [устройство] [точка_монтирования]
```

Назначение опций таково:

- `-a` — монтирование всех файловых систем, описанных в файле `/etc/fstab`, за исключением тех, что помечены ключевым словом `noauto` или уже смонтированы или исключены благодаря опции `-t`.
- `-d` — выполнение всех операций, связанных с монтированием, за исключением собственно монтирования. Используется для отладки, особенно в сочетании с опцией `-v`.
- `-f` — Эта опция имеет две интерпретации. Во-первых, принудительно монтируется непроверенная файловая система (те, некорректно демонтированная) в режиме чтения/записи, что небезопасно. Во-вторых, отменяется доступ в режиме чтения/записи к открытым файлам, если файловая система повторно монтируется в режиме "только чтение" с помощью опции `-u`.
- `-p` — вывод информации о смонтированных файловых системах в формате файла `/etc/fstab`.
- `-r` — монтирование файловой системы в режиме "только чтение".
- `-u` — изменение статуса смонтированной файловой системы. Например, команда `mount -ru /floppy` повторно монтирует уже смонтированную файловую систему `/floppy` в режиме "только чтение".
- `-v` — вывод информации о ходе монтирования.
- `-w` — монтирование файловой системы в режиме чтения/записи. Это установка по умолчанию для большинства файловых систем.
- `-o` *параметры*. С помощью опции `-o` можно менять способ обработки файловой системы. Среди наиболее распространенных параметров назовем `nodev` (игнорировать файлы устройств), `noexec` (не запускать исполняемые файлы), `nosuid` (игнорировать бит SUID для исполняемых файлов), `nosymfollow` (не переходить по символическим ссылкам) и `union` (при подключении к точке монтирования добавлять файлы к уже существующим, а не делать существующие файлы недоступными). Эти и другие параметры описаны на man-странице команды `mount`. Параметры, специфичные для различных файловых систем, рассматривались в главе 4.

- `-t` *тип* — задает тип файловой системы. Значение по умолчанию `ufs` Ему соответствует собственная файловая система FreeBSD: FFS. Другие возможные типы: `cd9660` (компакт диски формата ISO-9660, Joliet и Rock Ridge), `msdosfs` или `msdos` (дискеты и разделы DOS или Windows), `ext2fs` (разделы ext2fs в Linux), `ntfs` (NTFS-разделы в Windows NT/2000/XP) и `nfs` (файловые системы, экспортируемые по сети через NFS).
- *устройство* — имя файла устройства, связанного с разделом или съемным носителем В случае NFS здесь будет указана спецификация экспортируемой файловой системы, состоящая из имени сервера, двоеточил и пути к экспортируемому каталогу, например `nfsserver:/home`.
- *точка монтирования* — каталог через который должны быть доступны данные. Обычно он пуст.

Если файловая система описана в файле `/etc/fstab`, то команде `mount` достаточно указать либо имя устройства, либо точку монтирования. Все необходимые опции будут прочитаны из файла Такой подход часто применяется при работе со съемными носителями в частности дискетами и компакт-дисками Например следующая команда монтирует компакт-диск:

```
# mount /cdrom
```

Если для монтируемого устройства нет записи в файле `/etc/fstab`, задаите хотя бы файл устройства Предположим, инсталлируется новый жесткий диск, предназначенный для хранения каталога `/home`. Если диск отформатирован и на нем создана файловая система (устройство `/dev/ad2s1e`), то можно временно смонтировать его в точке `/mnt`.

```
# mount /dev/ad2s1e /mnt
```

Затем следует скопировать существующий нитвлог `/home` со всеми подкаталогами в новый раздел, удалить исходные файлы и отредвктировать файл `/etc/fstab`, чтобы новый раздел монтировался в точке `/home`.

Скопировав файлы на новый диск, не удаляйте исходные копии, пока новая конфигурация не будет протестирована. Если окажется, что какие-то файлы не скопированы корректно, их можно прочесть из исходного каталога. Переименуйте старый каталог с помощью команды `mv` или же демонтируйте старый раздел и не используйте его для других целей несколько часов или дней.

Если монтируются разделы или диски, не содержащие FFS, используйте команду `mount` совместно с опцией `t` Существуют также разновидности этой команды, предназначенные для работы с конкретными файловыми системами, например `mount_cd9660` и `mount_msdosfs`. Они описывались в главе 4

Файл /etc/fstab

Файл `/etc/fstab` определяет, какие разделы монтируются при загрузке FreeBSD Пример этого файла показан в листинге 7 1 Первая строка является комментарием (помечается символом '#') и содержит названия полей файла Поля разделяются пробелами или символами табуляции

Листинг 7.1 Образец фаила `/etc/fstab`

```
# Device        Mountpoint      FStype      Options         Dump    Pass#
/dev/ad0s2b     none            swap        sw                 0    0
/dev/ad0s2a     none            ufs         rw                 1    1
/dev/ad0s2f     none            ufs         rw                 2    2
```

```
/dev/ad0s      none            msdosfs  rw  s, u=1000    0    0
/dev/acd0c     none            cd9660   ro,noauto        0    0
proc           none            procfs   rw               0    0
speaker /home /speaker/home    nfs      rw               0    0
```

Большая часть информации, хранящейся в файле, указывается в качестве аргументов команды `mount`. В частности, в поле `Device` задается файл устройства, связанный с разделом или диском, в поле `Mountpoint` указана точка монтирования, а в поле `FStype` приведен тип файловой системы. Значение поля `Dump` говорит о том, как часто при использовании утилиты `dump` должно архивироваться содержимое раздела. В поле `Pass#` задан порядок проверки файловых систем командой `fsck` в процессе начальной загрузки. Обычно корневая файловая система (`/`) проверяется на первом проходе, остальные разделы FFS — на втором.

Поле `Options` требует более подробного объяснения. В нем перечислены параметры, передаваемые команде `mount` посредством опции `-o`. Некоторые из них дублируют действие других опций команды, остальные специфичны для опции `-o` или используются только в файле `/etc/fstab`. Есть также параметры, используемые для конкретных файловых систем. Ниже перечислены наиболее распространенные параметры:

- **`ro` и `rw`**. Эти параметры обозначают монтирование в режиме только для чтения и в режиме чтения/записи. Изначально для каждой файловой системы задан один из этих параметров.
- **`noauto`**. По умолчанию FreeBSD монтирует файловую систему в процессе начальной загрузки. Некоторые устройства, например дисководы CD-ROM, лучше конфигурировать с параметром `noauto`, чтобы на этапе начальной загрузки система их игнорировала. Для последующего монтирования устройства нужно выполнить команду вида **`mount /mnt/cdrom`**.
- **Параметры конкретных файловых систем**. В листинге 7.1 для точки монтирования `/windows` используются специальные параметры монтирования. В частности, параметры `-s` и `-u=1000` заставляют FreeBSD игнорировать длинные имена файлов VFAT и назначить владельцем всех файлов пользователя с идентификатором 1000. Эти и другие подобные опции описывались в главе 4. Дополнительную информацию о них можно получить на man-страницах соответствующих вариантов команды `mount`, например `mount_msdosfs`.

При создании файла `/etc/fstab` изменения не вступают в силу немедленно. Введите **`mount -a`** или перезагрузите компьютер, чтобы новые файловые системы были смонтированы. Удаленные файловые системы не исчезнут, пока не выполнить перезагрузку или не демонтировать их вручную.

В листинге 7.1 присутствует особая файловая система `proc`, которая смонтирована в каталоге `/proc`. Это не файловая система в общепринятом смысле, а специальный интерфейс ядра, позволяющий программам получать информацию об аппаратных устройствах компьютера. Подобно дисковым файловым системам, ее нужно монтировать. Она должна обязательно присутствовать в файле `/etc/fstab`, поскольку с ней работают многие программы.

В последней строке листинга 7.1 демонстрируется спецификация доступа к экспортируемой файловой системе NFS. Как и в случае команды `mount`, здесь указывается имя сервера и путь к каталогу на сервере. О серверах NFS будет рассказываться в главе 18.

Демонтироввние разделов

По окончании работы с дискетой, компакт-диском или другим съемным носителем нужно *демонтировать* его, прежде чем извлекать из устройства. Это заставит FreeBSD завершить все отложенные операции, а также пометить раздел как стабильный. Можно демонтировать и дисковый раздел, если требуется сделать его недоступным.

Дисководы гибких дисков на компьютерах семейства *x86* имеют кнопку принудительного извлечения дискеты. При нажатии такой кнопки демонтирование не выполняется. *Это может привести к повреждению данных!* Всегда демонтируйте дискету, прежде чем извлекать ее.

Команда демонтирования устройства называется `umount`. Она имеет следующий синтаксис:

```
umount [-fv] [ a | A]   h узел] [ t тип] устройство | точка монтирования
```

Назначение опций таково:

- **`-f`** — принудительное демонтирование, даже если на устройстве есть открытые файлы. Желательно избегать этого.
- **`v`** — вывод дополнительной информации о ходе демонтирования.
- **`-a`** — демонтирование всех файловых систем, описанных в файле `/etc/fstab`, за исключением корневой (`/`).
- **`-A`** — демонтирование всех смонтированных файловых систем, кроме корневой (`/`).
- **`h`** *`узел`*. Эта опция обычно используется для демонтирования файловых систем, экспортируемых с сервера NFS. Но в сочетании с опцией `t` она позволяет демонтировать локальные файловые системы.
- **`-t`** *`тип`* — демонтирование файловых систем заданного типа.

В большинстве случаев команду `umount` применяют для демонтирования отдельной файловой системы, указывая имя устройства или точку монтирования. Например, следующая команда демонтирует компакт-диск:

```
# umount /mnt/cdrom
```

Иногда требуются дополнительные опции. Например, опция `f` удобна, если пользователи продолжают работать с устройством, которое нужно демонтировать. Но она может привести к тому, что файлы исчезнут в процессе работы, а это чревато непредсказуемыми проблемами. Рассматривайте опцию `-f` как крайнее средство.

Традиционные схемы разбивки на разделы

Мы рассмотрели утилиты работы с разделами, сделав акцент на синтаксисе их использования. Как же применять эти утилиты, т.е. какие типы разделов создавать, где их монтировать и т.д.? Частично ответ на этот и другие вопросы можно получить, определив, какие типы файлов будут содержаться в том или ином разделе. Существуют также правила, традиционные для FreeBSD (или вообще для UNIX), и практические ограничения.

Какие каталоги могут быть разделами?

Первый вопрос, на который нужно ответить, звучит так: какие каталоги могут быть разделами? Ответ прост: большинство. Дерево каталогов FreeBSD располагается в

корневом разделе (/), который содержит как обычные каталоги так и точки монтирования. Корневому каталогу нужен отдельный раздел, и большинство его непосредственных подкаталогов тоже может содержаться в собственных разделах. Исключение составляют каталоги /etc, /sbin, /bin и /boot (отдельно создаются файловые системы /dev и /proc, но они не занимают дисковых разделов). В этих каталогах хранятся утилиты, необходимые для загрузки системы.

Иногда в отдельных разделах размещают и более глубокие подкаталоги, например /usr/local или /usr/X11R6. При этом не имеет значения, является ли каталог /usr разделом или обычным подкаталогом корневого раздела.

Распространенные схемы разбивки на разделы

Инсталлятор FreeBSD способен автоматически создавать стандартные конфигурации разделов. На основе параметров среднестатистической системы инсталлятор самостоятельно выбирает размеры корневого раздела (/), разделов /home, /usr, /tmp, /var и области подкачки. Об этом рассказывалось в главе 2. В каталоге /home хранятся пользовательские файлы, в каталоге /usr — большинство программных файлов, в каталоге /tmp — временные файлы, в каталоге /var — очереди и другие временные данные системного уровня. Область подкачки используется как расширение системного ОЗУ. При желании можно создать всего два раздела: корневой и область подкачки. Но чаще создают дополнительные разделы, например /usr/X11R6, /usr/local, /opt, или распределяют пользовательские каталоги между несколькими разделами, допустим /home и /home2.

Использование дополнительных разделов дает следующие преимущества:

- **Переменные параметры монтирования файловых систем.** В каждом разделе можно создавать различные файловые системы с разными параметрами монтирования, оптимизируя тем самым производительность системы и повышая ее безопасность. Например, раздел /usr можно смонтировать только для чтения, резко снизив риск модификации важных системных файлов.
- **Эффективное расположение файловых систем.** Тщательно все спланировав, можно разместить разделы на диске с учетом максимального повышения производительности. Часто используемые разделы должны располагаться ближе к центру диска для сокращения времени позиционирования головки диска. Этого можно добиться с помощью утилиты fdisk или disklabel. Если есть два физических диска, то два наиболее активно используемых раздела должны находиться на разных дисках, опять-таки в целях повышения производительности.
- **Повышенная надежность.** Иногда вследствие сбоев питания происходит потеря данных на диске. Если размер рабочих разделов невелик, будет потеряно меньше данных и упростится процедура восстановления.

Естественно, нельзя забывать и о недостатках небольших разделов. Чем меньше раздел, тем быстрее он переполняется, особенно если неправильно определен его размер. Это ограничение можно обойти с помощью символических ссылок на другие разделы, но такая конфигурация требует дополнительной настройки.

Монтирование внешних разделов

Помимо стандартных разделов FreeBSD может возникнуть необходимость смонтировать внешние файловые системы. Под таковыми подразумеваются разделы других операционных систем, например Windows, или съемные носители, в частности дискеты и компакт-диски. Существуют два подхода к их монтированию.

- **Монтирование к корню** Внешние файловые системы можно монтировать непосредственно к корневому каталогу. Например, в стандартной конфигурации предусмотрена точка монтирования `/cdrom` для компакт-дисков. Можно создать точку монтирования `/c` или `/windows` для подключения разделов Windows
- **Монтирование к каталогу** `/mnt` Каталог `/mnt` традиционно используется для монтирования съемных носителей и других файловых систем В стандартной конфигурации в нем нет подкаталогов, но их можно создать например `/mnt/floppy` — для дискет или `/mnt/cdrom` — для компакт-дисков

Оба варианта были продемонстрированы в листинге 7.1, где присутствуют записи для каталогов `/windows` и `/mnt/cdrom`. Каталог `/speaker/home` предназначен для хранения файловых систем экспортируемых с сервера `speaker`.

FHS

Как рассказывалось в главе 1, в истории UNIX было немало моментов, когда операционная система разделялась на конкурирующие "ветви", а удачные идеи переносились из одной реализации системы в другую Следствием этого стало появление различных схем хранения файлов. Вообще говоря, это считается недостатком UNIX в целом, поскольку человек, научившийся работать в одной версии системы, может столкнуться с проблемами в другой версии просто из-за того, что те же самые файлы хранятся в другом месте Частичным решением проблемы стало появление в середине 90-х годов FHS (Filesystem Hierarchy Standard — стандарт иерархии файловой системы) Это формальный стандарт использования каталогов и размещения важных системных файлов. Ниже мы рассмотрим его особенности чтобы понять, почему во FreeBSD конфигурационные, программные и пользовательские файлы оказываются в тех или иных каталогах

Роль FHS

FHS описан в документах, доступных по адресу `http://www.pathname.com/fhs/` На момент написания книги последняя версия стандарта имела номер 2 2 (выпущена в мае 2001 года). Ведутся работы над версией 2.3, но, скорее всего, рассмотренные ниже основные особенности стандарта останутся неизменными в последующих версиях.

В FHS вводятся два ключевых разграничения, влияющих на общую структуру каталогов

- **Статические и динамические файлы** *Статические* файлы меняются редко в отличие от *динамических* К примеру, исполняемые файлы считаются статическими, а пользовательские файлы данных — динамическими
- **Совместно используемые и локальные файлы** Бывают ситуации, когда работа с файлами должна вестись с нескольких компьютеров, но есть файлы, которые могут использоваться только локально. Например, пользовательские файлы данных и программные файлы могут быть совместно используемыми, а локальные конфигурационные файлы нет

На заметку Можно сконфигурировать сервер NFS на экспорт каталога, содержащего локальные файлы, но это не дает никаких преимуществ и к тому же рискованно с точки зрения безопасности

Совместно эти критерии определяют четыре типа каталогов (рис 7 3). Иногда ка талоги, расположенные непосредственно в корневом каталоге, нельзя отнести к кон кретному типу Цель классификации заключается в том, чтобы можно было легко оп ределить какие даниме следует хранить в разделах, доступных только для чтения, и отделить каталоги, допускающие экспорт через NFS, от тех, которые не могут экспор тироваться

	Совместно используемый	Локальный
Статический	/usr opt	/etc /boot
Динамический	home /var/mail	/var/run var log

Рис 7.3 Пример классификации каталогов

На заметку Приведенную классификацию нельзя считать абсолютиым правилом Например каталог /home помещен на рис. 7.3 в категорию "Совме стно используемый", но далеко не всегда он делается экспортируемым

В рамках представленной классификации FHS выделяет ряд обязательных и не обязательных каталогов Для того чтобы операционная система была совместима со стандартом, она должна содержать обязательные каталоги или по крайней мере сим волические ссылки с соответствующими именами В стандарте приведена также до полнительная информация о некоторых каталогах К примеру, требуется, чтобы ката лог /bin не содержал подкаталогов, а для ряда каталогов определен минимальный на бор файлов

Обзор кталогов FHS

Спецификация FHS представляет собои 41-страничный документ поэтому стан дарт невозможно полностью описать в этой главе Рассмотрим ключевые каталоги определенные в стандарте

- **Корневой (/)** — начальная точка файловой системы Согласно стандарту в кор невом каталоге должны содержаться подкаталоги или точки монтирования /bin, /boot, /dev /etc, /lib, /mnt, /opt, /sbin, /tmp, /usr и /var. Преду смотрены также три необязательных каталога присутствующих во многих сис темах /home, /lib<*тип*> и , root.
- **/bin** — статический каталог, допускающий совместный доступ но он редко монтируется клиентами так как почти всегда присутствует в системе В нем находятся основные программы пользовательского уровня, такие как ls, cp и sh. Идея заключается в том чтобы эти команды были доступны даже когда некорневые разделы не удается смонтировать
- **/boot** — статический локальный каталог, который хранит основные загрузоч ные файлы, в частности файл ядра
- **/dev** — статический локальный каталог, хранящий фаилы устройств, через ко торые осуществляется доступ к аппаратным устройствам компьютера. Во FreeBSD 5 0 такой доступ реализуется с помощью специальной файловой сис темы devfs, которая генерирует файлы для обнаруженных устройств
- **/etc** — статический и локальный каталог Его содержимое было описано в главе 5 Напомним что в нем хранятся файлы конфигурации системы

- **/lib** – статический каталог, допускающий совместный доступ, но, как и /bin, экспортируется редко. В нем хранятся важнейшие системные библиотеки, используемые программами из каталогов /bin и /sbin. Во FreeBSD каталог /lib отсутствует поскольку упомянутые программы компонуются *статически* и для их работы не требуются внешние библиотеки
- **/mnt** – точка монтирования временных файловых систем В стандартной инсталляции FreeBSD каталог /mnt не содержит подкаталогов, но некоторые администраторы создают подкаталоги наподобие /mnt/floppy и /mnt/cdrom для монтирования соответствующих устройств
- **/opt** — статический каталог, допускающий совместный доступ и предназначенный для хранения вспомогательных программных пакетов. Но на практике во FreeBSD большинство таких пакетов находится в каталоге /usr/local, поэтому каталог /opt используется мало Названия его подкаталогов соответствуют названиям програмы, например /opt/OpenOffice.org
- **/sbin** — аналог каталога /bin, содержащий команды предназначенные для системного администратора, например shutdown и fsck
- **/tmp** — динамический локальный каталог, который хранит временные файлы Он должен быть доступен для чтения/записи всем пользователям и программам. Программа не должна предполагать, что находящиеся в нем файлы остануться доступными после ее завершения хотя на практике часто именно так и происходит
- **/usr** — статический каталог, допускающий совместный доступ. Он очень велик, поскольку хранит большинство программных файлов системы. В нем есть подкаталоги, аналогичные уже упоминавшимся выше, например /usr/bin /usr/sbin и /usr/lib, но в них находятся обычные программы, не участвующие в обеспечении базовых функциональных возможностей системы. Некоторые из подкаталогов представляют особую важность и потому описаны отдельно.
- **/usr/local** — в этом подкаталоге находятся локальные файлы, созданные или инсталлированные специально для этой системы, в отличие от файлов, составляющих стандартную часть операционной системы. Например, если программа компилируется "с нуля", а не инсталлируется в готовом виде, нужно защитить ее от процедур автоматического обновления системы Каталог /usr/local активно используется во FreeBSD, в нем инсталлируется множество пакетов включая большинство программ из коллекции переносимых приложений (методы инсталляции программ рассматриваются в главе 11).
- **/usr/X11R6** — в этом подкаталоге хранятся файлы X Window System включал программный код X-серверов.
- **/var** — в этом каталоге содержатся динамические файлы данных, а подкаталоги могут быть как локальными, так и совместно используемыми. Здесь хранятся идентификаторы процессов (/var/run), почтовые буферы (/var/mail) системные журнальные файлы (/var/log) и т.д.
- **/home** – необязательный каталог предназначенный в основном для хранения пользовательских файлов данных Редко какая система обходится без каталога /home, хотя иногда он называется по-другому или разбивается на несколько каталогов. Как правило, у каждого пользователя в этом каталоге есть свой индивидуальный подкаталог, но могут быть и подкаталоги общие для всех пользователей.
- **/lib**<*тип*> — в стандарте этот необязательный каталог обозначен как статический, допускающий совместный доступ. В качестве компонента <*тип*> задается

тип библиотеки. Идея заключается в том, чтобы система могла иметь несколько библиотечных каталогов, каждый из которых обеспечивает поддержку определенного набора системных возможностей. Чаще всего это используется в 64-разрядных платформах для поддержки 32- и 64-разрядных приложений. Следовательно, если FreeBSD устанавливается на компьютерах семейства x86, каталоги `/lib<тип>` отсутствуют.

- **`/root`** — это необязательный динамический локальный каталог, являющийся начальным каталогом суперпользователя. В нем содержатся суперпользовательские конфигурационные файлы и другие файлы, которые суперпользователю приходится генерировать и/или хранить отдельно. Если каталог `/root` не существует, начальным каталогом суперпользователя становится корневой (`/`).

Дополнительная информация о стандартных каталогах доступна на Web-узле FHS (`http://www.pathname.com/fhs`). Сведения, касающиеся непосредственно FreeBSD, можно получить с помощью команды `man hier`.

Подводя итог, отметим, что FHS определяет файловую структуру UNIX-подобных операционных систем, задавая расположение важных групп файлов, в частности, библиотечных, программных, административных, конфигурационных, временных пользовательских и др. Знание этой структуры помогает находить нужные файлы.

Уровень совместимости с FHS

FreeBSD практически полностью совместима с FHS, содержит все обязательные каталоги и в целом следует стандарту, хотя есть и некоторые несоответствия. К примеру, в стандарте указано, что программы `login` и `mount` должны находиться в каталоге `/bin`, тогда как во FreeBSD они помещены в каталоги `/usr/bin` и `/sbin` соответственно. Кроме того, во FreeBSD широко используются каталоги, не включенные в стандарт (они ни в коем случае не нарушают стандарт, просто не упомянуты в нем явно). В частности, в стандарте ничего не говорится о каталоге `/usr/local/etc`, а во FreeBSD многие программы записывают в него свои конфигурационные файлы. Таким образом, FHS представляет собой ценное руководство, важное для понимания структуры каталогов FreeBSD, но его нельзя рассматривать как стопроцентно точное описание системы.

Резюме

Управление дисковыми разделами — одна из важнейших задач системного администратора. Первоначально разделы создаются при инсталляции системы. Необходимость в их модификации или создании новых разделов возникает при добавлении или замене жесткого диска либо в том случае, если исходная схема разбивки на разделы перестает удовлетворять потребностям пользователей. Разделы создаются и на некоторых съемных носителях, например на Zip-дисках. Во FreeBSD существует утилита `sysinstall`, реализующая удобный интерактивный интерфейс управления разделами и файловыми системами. Те, кто предпочитают работать в режиме командной строки, могут вызывать соответствующие низкоуровневые утилиты (`fdisk`, `disklabel` и `newfs`) напрямую.

Созданный раздел монтируется с помощью команды `mount`, которая подключает его файловую систему к дереву каталогов системы. Некоторые разделы можно монтировать где угодно, но структура каталогов FreeBSD требует, чтобы определенные файлы находились в строго определенных каталогах. Понимание этой структуры важно для монтирования новых разделов или съемных носителей.

Полный справочник по

Как администратор, так и пользователи FreeBSD должны уметь управлять файлами. В конце концов работа за компьютером в той или иной степени сводится к работе с файлами. В этой главе рассказывается о средствах управления файлами во FreeBSD. Для начала мы поговорим о правах доступа к файлам и принципах владения файлами. Затем будут рассмотрены многочисленные команды, предназначенные для манипулирования файлами, в том числе для копирования, удаления, поиска файлов, изменения их владельцев и прав доступа. В конце главы приведена информация о резервном копировании данных.

На заметку

В этой главе представлены команды текстового режима. Те же самые действия можно делать с помощью графических диспетчеров файлов, но они гораздо менее стандартизированы, к тому же существует множество конкурирующих диспетчеров файлов. В главе 3 кратко описывались средства работы с файлами в графической оболочке GNOME.

Права доступа и принадлежность файлов

FreeBSD, будучи одной из разновидностей UNIX, основана на концепциях прав доступа к файлам и принадлежности файлов. С этими двумя концепциями связаны многие средства защиты системы, и именно они позволяют множеству пользователей работать за одним компьютером, не беспокоясь по поводу случайного или преднамеренного повреждения своих файлов кем-либо. У каждого файла есть владелец и набор прав доступа, разрешающих или запрещающих доступ к файлу. Понимание механизмов доступа чрезвычайно важно для эффективного управления файлами. Особенно это касается системного администратора, задача которого — установить надлежащие права доступа к системным файлам и каталогам.

Учетные записи и владельцы файлов

Как было сказано в главах 2 и 3, учетная запись представляет собой набор взаимосвязанных структур данных для доступа в систему. Учетные записи могут служить и внутрисистемным целям, например быть связанными с конкретным сервером. С одной учетной записью могут работать сразу несколько пользователей, хотя так поступать не рекомендуется, поскольку это усложняет административный контроль.

Каждая учетная запись связана с одной или несколькими *группами*. Под группой понимается именованный набор учетных записей. Предположим, в системе зарегистрированы четыре пользователя: `dana`, `fox`, `walter` и `monica`. Они могут быть распределены на две группы: `research` и `development`. В первую входят пользователи `dana` и `walter`, во вторую — `dana`, `fox` и `monica`. В каждой системе есть также несколько стандартных групп, предназначенных для служебных учетных записей.

Каждый файл принадлежит одному пользователю и одной группе. Информация о владельцах файла хранится в виде *идентификатора пользователя* (User ID — UID) и *идентификатора группы* (Group ID — GID). Эти идентификаторы выделяются при создании учетной записи пользователя или группы. К примеру, файлу могут быть сопоставлены идентификатор пользователя 1021 и идентификатор группы 1003. Этим значениям, в свою очередь, могут соответствовать пользователь `dana` и группа `development` (вообще говоря, числовые идентификаторы являются внутренним средством самой системы, тогда как символические имена наподобие `dana` и `development` применяются при взаимодействии с пользователем).

Пользователь, создающий файл, становится его владельцем. В качестве группы выбирается группа того каталога, в котором создается файл. Например, если каталог `/home/dana` принадлежит группе `research`, то создаваемые в нем файлы будут принадлежать этой же группе. Если пользователь `dana` хочет создать файлы, принадлежащие группе `development`, он должен воспользоваться командой `chgrp` (описывается далее) или создать подкаталог, поменять его группового владельца с помощью команды `chgrp` и уже в нем создавать новые файлы.

С понятиями группового и индивидуального владения файлами связан ряд нюансов, которые в определенных ситуациях играют важную роль.

- **Отсутствие связи между владельцем файла и группой, которой он принадлежит.** Пользователь не обязан входить в группу, которой принадлежит файл. Например, владельцами файла могут быть пользователь `walter` и группа `development` несмотря на то что пользователь `walter` не является членом этой группы.
- **Соответствие между идентификаторами пользователей в различных системах.** Даже если у пользователя есть учетные записи в нескольких системах, у них могут быть разные идентификаторы. Это несовпадение может привести к тому, что, когда файл переносится из одной системы в другую (на дискете с файловой системой формата FFS или средствами NFS), у него внезапно меняется владелец. Точно так же, если на компьютере установлено несколько различных UNIX-систем, нужно сделать так, чтобы одним и тем же учетным записям соответствовали одинаковые идентификаторы. Это позволит избежать проблем при чтении файлов в другой системе.
- **Изменение идентификатора пользователя.** Если по какой-то причине меняется идентификатор учетной записи пользователя, нужно поменять владельца всех файлов, принадлежащих этому пользователю.

Учетные записи связаны не только с файлами. Каждому запускаемому процессу тоже назначаются идентификаторы пользователя и группы. В общем-то, именно они должны соответствовать аналогичным идентификаторам файлов, к которым осуществляется доступ. Когда пользователь регистрируется в системе, FreeBSD запускает интерпретатор команд, назначая ему значения UID и GID этого пользователя.

Контроль доступа к файлам

Войдите в систему в текстовом режиме и введите **`ls -l`**. В любом непустом каталоге команда отобразит одну или несколько строк примерно такого вида:

```
-rw-r--r--  1 fox      users   517 Jan 31 21:23  file.txt
```

Это образец расширенного листинга команды. Здесь указаны имя файла (`file.txt`), дата и время его создания (`Jan 31 21:23`), размер (517 байтов), группа, которой он принадлежит (`users`), владелец (`fox`), число жестких ссылок (1; о ссылках на файлы рассказывается на с. 205) и права доступа (`-rw-r--r--`). Строка прав доступа заслуживает отдельного пояснения. Она состоит из десяти символов. Первый (крайний слева) определяет тип файла (табл. 8.1). Пользователи чаще всего приходится иметь дело с каталогами, обычными файлами и символическими ссылками. Файлы других типов встречаются реже, хотя по-своему важны.

Следующие девять символов в строке прав доступа разбиваются на три группы (триплеты) по три символа в каждой. Эти группы определяют права владельца файла, группы, которой он принадлежит, и остальных пользователей. В каждой группе задаются, слева направо, права чтения (`r`), записи (`w`) и выполнения (`x`). Если вместо указанного символа стоит дефис (`-`), соответствующая привилегия отсутствует. В некоторых случаях, о чем пойдет речь ниже, в поле выполнения стоит другой символ вместо `x`.

Таблица 8 1 Символические обозначения типов файлов

Символ	*Интерпретация*
	Обычный файл данных
d	Каталог Каталоги — это те же файлы хранящие имена других файлов и ссылки на них
l	Символическая ссылка. Это файл, хранящий имя другого файла Когда программа обращается к символической ссылке, она считывает данные из указанного файла
p	Именованный канал. Это средство взаимодействия двух программ одновременно открывающих один и тот же файл одна — для чтения, другая — для записи
s	Сокет Сокеты напоминают именованные каналы но используются для организации сетевых и двунаправленных соединений
b	Файл блочного устройства Файлы этого типа используются для доступа к устройствам которые поддерживают блочный обмен данными например жестким дискам
c	Файл символьного устройства Файлы этого типа используются для доступа к устройствам которые поддерживают посимвольный обмен данными, например последовательным портам

На заметку

Во FreeBSD программные файлы — как исполняемые, так и сценарии — идентифицируются по наличию права выполнения. Совсем по-другому обстоит дело в Windows, где для этих же целей используются расширения файлов, в частности .EXE и .COM. Исполняемые файлы FreeBSD обычно не имеют таких расширений, хотя они могут присутствовать.

Другой способ представления прав доступа — использование восьмеричного кода Каждый символ в группе считается битом трехразрядного числа, поэтому строке прав доступа соответствует трехзначное восьмеричное число Право чтения — это старший бит (восьмеричное 4), право записи — средний (восьмеричное 2), право выполнения — младший (восьмеричное 1) Если сложить эти значения, получится восьмеричный код группы К примеру, строка `rwxr-x---` преобразуется так 1) владелец имеет все права, поэтому получаем 4+2+1=7, 2) группе предоставлены права чтения и выполнения, но не записи поэтому 4+1=5, 3) остальные пользователи не имеют прав (0). Таким образом, строку `rwxr-x---` можно выразить восьмеричным числом 750 В табл. 8 2 приведен еще ряд примеров.

Таблица 8.2. Коды прав доступа и их интерпретация

Строка прав доступа	*Восьмеричный код*	*Интерпретация*
`rwxrwxrwx`	777	Владелец, члены группы и остальные пользователи имеют полный доступ к файлу
`rwxr xr x`	755	Владелец имеет полный доступ к файлу члены группы и остальные пользователи могут читать и выполнять файл, но не записывать в него
`rwxr x`	750	Владелец, имеет полный доступ к файлу члены группы могут читать и выполнять файл но не записывать в него. Остальным пользователям доступ запрещен
`rwx x x`	711	Владелец имеет полный доступ к файлу, члены группы и остальные пользователи могут выполнять файл, но не осуществлять чтение/запись
`rw rw rw`	666	Владелец, члены группы и остальные пользователи могут осуществлять чтение/запись файла, но не могут выполнять его
`rw r r`	644	Владельцу разрешены чтение/запись файла члены группы и остальные пользователи могут только читать файл
`r`	400	Владелец, может только читать файл, членам группы и остальным пользователям доступ запрещен

На заметку

Всего существует 2^9 т.е 512 возможных комбинаций битов для 9 разрядных строк прав доступа

Теоретически члены группы могут иметь меньше прав чем остальные пользователи Например, режим доступа `rw----r--` (604) формально означает, что остальные пользователи имеют право чтения, а члены группы — нет Впрочем, на практике этого ограничения не существует, поскольку категория "остальные пользователи" охватывает и членов группы тоже, т.е. указанный выше режим доступа эквивалентен режиму `rw-r--r--` (644) Что касается прав владельца и группы, то владелец может не быть членом группы, которой принадлежит файл поэтому допускается ситуация когда члены группы имеют больше прав, чем владелец файла. Но такую конфигурацию нельзя назвать типичной, и она вряд ли имеет практическое применение.

Символические ссылки обычно имеют режим доступа `rwxr-xr-x` (755) Это означает, что пользователи могут узнать имя файла на который указывает ссылка У самого файла могут быть другие права доступа.

Важным исключением из правил является учетная запись `root`. Суперпользователю разрешено осуществлять чтение и запись любого файла системы независимо от того, кто его владелец и каковы права доступа к файлу. Суперпользователь может даже работать с файлом, у которого режим доступа 000 (`-------`). Помимо этого суперпользователь имеет право менять владельца любого файла и задавать произвольные атрибуты доступа Такие возможности нужны системному администратору, чтобы держать систему под контролем

В некоторых относительно новых конфигурациях пользователь `root` не имеет полного доступа ко всем файлам Это может быть очень выгодно с точки зрения безопасности, поскольку есть серверы, работающие в режиме суперпользователя, что дает им — и их ошибкам — полную власть над системой. Другое ограничение возможностей суперпользователя связано с доступом к сторонним файловым системам Например, когда во FreeBSD монтируется раздел формата FAT или дискета, всем файлам назначаются один и тот же владелец и одинаковые права доступа (лишь атрибут "только чтение" учитывается как инвертированный бит права записи). Даже суперпользователь не может изменять эти атрибуты (за исключением права записи) на уровне отдельных файлов, потому что в FAT это не предусмотрено. Существует лишь команда `mount_msdosfs` (см главу 4), работающая на уровне разделов FAT

Помимо рассмотренных выше стандартных прав доступа существуют три специальных бита, контролирующих доступ к исполняемым файлам и каталогам

- **SUID (Set User ID — смена идентификатора пользователя).** Этот бит может устанавливаться для исполняемых файлов Обычно программе назначается идентификатор пользователя, запустившего ее. Если же для нее установлен бит SUID, то используется идентификатор владельца программного файла. Бит SUID находит применение в программах, для работы которых требуются привилегии суперпользователя. Такие программы потенциально опасны, но иногда это единственная возможность предоставить рядовым пользователям доступ к привилегированным файлам. У программ с установленным битом SUID право выполнения для владельца обозначается как `s`, а не `x`, например `rwsr-xr-x`
- **SGID (Set Group ID — смена идентификатора группы)** Этот бит во многом аналогичен биту SUID, но относится к идентификатору группы Программа, у которой установлен бит SGID, в момент запуска принимает значение GID своего программного файла, а не текущего пользователя. У программ с установленным битом SGID право выполнения для группы обозначается как `s` а не `x` например `rwxr-sr-x`

- **Sticky-бит** В современных версиях FreeBSD так называемый *sticky bit* ("sticky' — липкий) применяется только к каталогам указывая на то что удалять файлы в каталоге разрешено лишь их владельцам либо владельцу самого каталога (в противном случае пользователь, имеющий право записи в каталог, сможет удалять чужие файлы). Чаще всего этот бит устанавливается для общедоступных каталогов `/tmp` и `/var/tmp` У каталогов с установленным sticky-битом право выполнения для остальных пользователей обозначается как `t` а не `x`, например `drwxr-xr-t`.

Перечисленные специальные биты не должны использоваться бездумно Как правило, они устанавливаются лишь в том случае, если в документации к программе сказано, что тот или иной бит действительно необходим для нормальной работы

Особенности доступа к каталогам

Те же права доступа применительно к каталогам приобретают несколько иной смысл Но прежде всего нужно понять, что каталог — это тоже файл но особого вида В нем хранятся имена каталожных файлов и указатели на индексные дескрипторы которые в свою очередь ссылаются на сектора диска, где хранятся данные. Как и любому другому файлу, каталогу назначаются владелец и права доступа. Таким образом, если пользователь имеет право записи в каталог, он может добавлять и удалять файлы в нем, *даже не будучи владельцем удаляемого файла*! Вот почему запись в начальные каталоги пользователей почти всегда разрешена только их владельцам. Этим объясняется и наличие sticky-бита у общедоступных каталогов наподобие `/tmp`. Без него каталоги, используемые для совместного хранения временных файлов или обмена файлами не были бы защищены от случайного или преднамеренного повреждения

Каталог — это не исполняемый файл, поэтому биты, обозначающие право выполнения, в случае каталогов трактуются по-другому. Они указывают ия возможность поиска информации в каталоге. Пользователь, не имеющий права поиска в каталоге, может выполнить команду `ls`, чтобы получить простой список содержимого каталога, но попытка найти в нем файл с помощью команды `find` (описана далее в главе) потерпит неудачу Даже некоторые опции команды `ls`, например `-l` (отображение прав доступа к файлу, даты его создания и т.д.), окажутся заблокированными

Разработка общесистемных правил доступа

Одна из наиболее важных задач администратора FreeBSD заключается в разработке правил доступа, общих для всех пользователей системы В ходе инсталляции системы рядовым пользователям разрешается доступ для чтения к большинству каталогов. Пользователи должны иметь такой доступ к каталогам, в которых хранятся основные программы, например `/bin` и `/usr/bin` Но некоторые каталоги, такие как `/root` должны оставаться закрытыми от посторонних глаз. Ряд каталогов, в частности `/tmp` следует делать доступными как для чтения, так и для записи. Ниже перечислены изменения, которые чаще всего вносятся в стандартную конфигурацию

- **Ограничение доступа к системным каталогам** Следует запретить рядовым пользователям доступ к важным каталогам. Например, можно создать группу `xusers`, закрепить ее за каталогом `/usr/X11R6` и сбросить для него все биты последнего триплета. Тогда запуск X-программ будет разрешен только членам группы `xusers` Часто контроль доступа ведется на уровне отдельных программ, а не каталогов. Недостатком такого подхода является то что большинство программ FreeBSD доступно в Internet и пользователи могут легко обойти многие ограничения создавая дубликаты программ в своих начальных каталогах

- **Совместно используемые каталоги** Если пользователи компьютера совместно работают над различными проектами, то общие файлы можно размещать в каталогах совместного доступа Закрепите такие каталоги за соответствующими группами, чтобы контролировать, кто имеет доступ к каталогу Например, если каталоги `/home/projects/research` и `/home/projects/development` принадлежат группам `research` и `development` и для обоих задан режим доступа `rwxrwx` (770), только члены указанных групп смогут осуществлять чтение и запись в этих каталогах Пользователи входящие и в ту и в другую группу получат доступ к обоим каталогам
- **Права доступа к начальным каталогам** Права доступа к начальным каталогам пользователей должны соответствовать принципам разграничения доступа принятым в организации По умолчанию выбирается режим доступа `rwxr xr-x` (755). Где любой пользователь может перейти в каталог другого пользователя и просмотреть его содержимое Более правильная установка — `rwxr-x---` (750) или даже `rwx -----` (700). Выбор зависит от того, какого типа группы создаются в системе (об этом рассказывается в главе 10)

Внимание! Пользователям разрешено менять права доступа к своим начальным каталогам. А это значит, что системы с высокими требованиями к безопасности должны регулярно инспектироваться, чтобы пользователи случайно или намеренно не поставили под удар собственные каталоги Есть и другие способы защиты Один из них — создать для каждого пользователя отдельную приватную группу, сделать владельцем всех начальных каталогов пользователя `root` и установить для них режим доступа `rwxrwx` - (770)

Команды манипулирования файлами

Изложенные выше сведения о правах доступа и владельцах файлов необходимы для понимания того, как работают команды манипулирования файлами В этом разделе описываются сами команды начиная с команд, позволяющих менять владельцев файлов и права доступа к ним. Далее рассматриваются средства копирования и удаления файлов, а также команды работы с символическими ссылками Здесь же рассказывается о командах поиска файлов по имени, содержимому или другим характеристикам

Изменение владельца файла

Первоначально файл принадлежит пользователю, который его создал Но бывают ситуации, когда требуется поменять владельца файла Например, если суперпользователь копирует созданный им конфигурационный файл в начальный каталог рядового пользователя, он должен передать последнему право владения файлом. Для этого предназначена команда `chown`, синтаксис которой таков.

```
chown [-fhv] [-R [-H | -L | -P]] владелец[:группа] файл[ ]
```

Ниже описано назначение опций команды

- **-f**. Обычно команда сообщает о неудачных попытках поменять владельца (файловая система смонтирована только для чтения, нет соответствующих прав и т.д.). Эта опция подавляет вывод таких сообщений.

- `h`. Когда команда применяется к символической ссылке, она обычно меняет владельца адресуемого файла. Эта опция заставляет команду поменять владельца самой символической ссылки.
- `-v`. Эта опция заставляет команду отображать имя каждого модифицируемого файла.
- `R`. Если в качестве аргумента указан не файл, а каталог, эта опция вызывает *рекурсивные* изменения, меняя владельца не только самого каталога, но также всех файлов в нем, всех подкаталогов и их содержимого. Опции `H`, `L` и `P` задают порядок обработки символических ссылок на каталоги. Опция `H` вызывает рекурсивный проход по символической ссылке, указанной в качестве аргумента. Опция `-L` вызывает рекурсивный проход по всем символическим ссылкам в заданном дереве каталогов. Опция `P` запрещает рекурсивную обработку символических ссылок.

В качестве аргумента команды `chown` задается новый владелец файла и/или группа (указывается через двоеточие). Вместо символических имен могут быть указаны идентификаторы (UID и GID). Должен присутствовать хотя бы один из этих аргументов, иначе команда не выполнится.

Рассмотрим примеры.

```
# chown walter csm.sxw
# chown dana:research *.tex
# chown -R :development /home/development
```

Первая команда делает владельцем одного файла (`csm.sxw`) пользователя `walter`. Вторая команда делает пользователя `dana` и группу `research` владельцами всех файлов текущего каталога, имеющих расширение `.tex`. В последнем примере группа `development` закрепляется за всеми файлами каталога `/home/development`, при этом символические ссылки не отслеживаются и владельцы файлов не меняются. Последний пример можно записать с помощью схожей команды `chgrp`, которая меняет только группу файла (перед названием группы не нужно указывать двоеточие). Вот эквивалентный вариант команды:

```
# chgrp -R development /home/development
```

Применение команд `chown` и `chgrp` для рядовых пользователей ограничено. В частности, пользователь может назначить файлу только ту группу, членом которой является он сам, и не может менять владельца чужих файлов.

Изменение прав доступа

Суперпользователю и владельцу файла разрешено менять права доступа к нему с помощью команды `chmod`. Ее синтаксис напоминает синтаксис команды `chown`:

```
chmod [-fhv] [-R [ H | L | -P]] режим файл[ ..]
```

Большинство опций имеет то же значение, что и в команде `chown`. Исключением является параметр *режим*, который представляет собой спецификацию прав доступа к файлу. Поддерживается два формата спецификаций: числовой и символический.

В числовой спецификации режим доступа задается в виде восьмеричного числа, как описывалось выше. Например, числу 644 соответствует спецификация `rw-r--r--`. Вместо трехзначной спецификации допускается четырехзначная. Дополнительная начальная цифра указывает на наличие специальных битов доступа. Начальная цифра 0 игнорируется, цифра 1 устанавливает sticky-бит, цифра 2 — бит SGID, цифра 4 — бит SUID. Эти значения можно складывать, включая несколько битов одновременно. Например, значение 6 соответствует битам SUID и SGID. Таким образом, число 6755 за

дает режим доступа `rwsr-sr-x`. Числовая спецификация удобна, когда известны все права доступа и особенно когда нужно поменять значения сразу нескольких битов.

В символической спецификации пользователь указывает, чьи конкретно права нужно поменять, каков тип изменения и какие права доступа должны быть установлены. Элементы спецификации описаны в табл. 8.3.

Таблица 8.3. Компоненты символической спецификации

Чьи права	Описание	Тип изменения	Описание	Права доступа	Описание
a	Все триплеты	=	Присваивание	r	Чтение
u	Владелец	+	Добавление	w	Запись
g	Группа		Удаление	x	Выполнение
o	Остальные пользователи			X	Установить бит выполнения, только если объект является каталогом или для файла уже установлен хотя бы один аналогичный бит
				s	Бит SUID или SGID
				t	Sticky-бит
				u	Существующие права для [illegible]
				g	Существующие права группы
				o	Существующие права остальных пользователей

Чтобы сформировать символическую спецификацию, нужно последовательно соединить элементы первой, третьей и пятой колонок таблицы. Допускается задавать несколько спецификаций, разделяя их запятыми. Примеры представлены в табл. 8.4, где показано, какими были первоначальные права и какими они стали в результате изменений.

Таблица 8.4. Примеры символических спецификаций

Начальные права	Спецификация	Измененные права
`rw-r--r--`	`a+x`	`rwxr-xr-x`
`rw-r--r--`	`og-r`	`rw-------`
`rw-r-----`	`gu=rwx`	`rwxrwx---`
`rw-r--r`	`g=u,o-r`	`rw-rw---`

У символических спецификаций есть определенные преимущества. Можно менять лишь выбранные права доступа, не затрагивая остальные, особенно когда они неизвестны. К примеру, если ввести `chmod -R a+rX` *каталог*, то заданный каталог станет доступным всем пользователям — они получат право чтения всех файлов в каталоге. Биты выполнения будут установлены только для самого каталога, его подкаталогов и тех файлов, у которых уже установлен хотя бы один такой бит. Что касается права записи, то никаких изменений не происходит. Эквивалентной числовой спецификации не существует.

Разрешается также менять стандартные права доступа, назначаемые создаваемым файлам. Для этого существует команда `umask`, которая встроена в большинство интерпретаторов и задает *пользовательскую маску доступа* (значение `umask`). Это восьме-

ричное число, определяющее биты доступа, которые *не должны* устанавливаться. По умолчанию выбирается маска 022. Она означает, что владельцу предоставляется полный доступ, а членам группы и остальным пользователям запрещена запись в файлы (помимо значения umask есть еще начальный режим доступа, который у файлов равен 666, поэтому пользователи по умолчанию не имеют права выполнять файлы). Вот как можно поменять значение umask:

```
$ umask 027
```

В этом примере посторонним пользователям запрещается доступ к создаваемым файлам. Эту команду можно добавить в сценарии регистрации пользователя либо в общесистемные конфигурационные файлы (см. главу 6)

Копирование и перемещение файлов

В главе 3 уже рассматривался ряд команд работы с файлами, включая cp и mv. Ниже будет подробнее описан синтаксис этих команд. Кроме того, будут представлены две дополнительные команды: cat и dd. Все это удобные средства для различного рода перемещений файлов.

Команда cp

Команда cp является основным инструментом копирования файлов во FreeBSD. Она поддерживает два формата вызова:

```
cp [-R [ H |  L | -P]] [ f | -i] [-pv] источник приемник
cp [-R [ H | -L |  P]] [ f  -i] [-pv] источник[.. ] целевой_каталог
```

В первом варианте команда копирует одиночный файл под другим именем в тот же каталог или под выбранным именем в другой каталог. Во втором варианте команда копирует один или несколько файлов в другой каталог с сохранением имен. Назначение опций команды таково.

- **-R** — включает режим рекурсивного копирования каталогов. Дополнительные опции H, L и P интерпретируются так же, как и в команде chown.
- **f** — заставляет команду перезаписывать существующие файлы без предупреждения в случае конфликта имен.
- **-i** — заставляет команду выдавать запрос (в поток stderr) при попытке перезаписи существующего файла.
- **p** Обычно при копировании файла не сохраняется информация о владельцах, правах доступа и дате создания. Опция -p заставляет команду сохранять основные характеристики файла, если у самого пользователя есть соответствующие полномочия (к примеру, только суперпользователь может менять владельца копии).
- **-v** — заставляет команду отображать имя каждого копируемого файла.

В качестве примера рассмотрим следующие команды:

```
$ cp file-2139.tex insects.tex
$ cp -Rp important-cases /mnt/zip
```

Первая команда создает в текущем каталоге копию файла file-2139.tex под именем insects.tex (копия может быть создана в другом каталоге, если к имени добавить спецификацию каталога, например /home/research/insects.tex). Вторая команда копирует все содержимое каталога important-cases в каталог /mnt/zip. Если important-cases — файл, будет скопирован только он.

На заметку

Групповое копирование файлов можно также выполнять с помощью команды tar, которая описывается далее в главе. Вообще говоря, команда tar является более надежным средством сохранения определенных характеристик файлов, поэтому ее предпочтительнее использовать для создания максимально точных копий каталогов например, при перемещении файлов на новый жесткий диск.

Команда mv

Команда mv синтаксически и функционально напоминает команду cp, но не поддерживает рекурсивный режим Она перемещает, а не копирует файлы Ее используют и для переименования файлов, т е перемещения их под другим именем в том же каталоге Синтаксис команды таков

```
mv [-f  -i] [-v] источник приемник
mv [-f  -i] [-v] источник[...] целевой_каталог
```

Назначение опций и аргументов команды такое же, как и у команды cp Иногда команда mv работает как cp, копируя файлы, только по окончании копирования исходные копии удаляются Это происходит в том случае, когда целевой каталог расположен в другой файловой системе, например на смонтированной дискете или Zip-диске. Если же файлы перемещаются в рамках одной файловой системы FreeBSD просто создает новую запись в целевом каталоге, а затем удаляет соответствующую запись из исходного каталога Данные в этом случае не перемещаются. Такая методика позволяет очень быстро перемещать файлы в пределах файловой системы независимо от их размеров.

Команда cat

Команда cat позволяет объединять несколько файлов в один. Часто она используется для просмотра текстовых файлов Синтаксис команды таков:

```
cat [-bnvtesu] [файл [.  ]]
```

Команда направляет данные в стандартный выходной поток (stdout), поэтому для конкатенации файлов необходимо использовать операторы переадресации (см гла ву 5) Ниже описано назначение опций команды

- b — включает нумерацию отображаемых строк (пустые строки не нумеруются)
- n – то же, что и предыдущее, но пустые строки тоже нумеруются
- v — заставляет команду отображать нелечатаемые символы в виде управляющих кодов, например ^X вместо <Ctrl+X>
- -t — то же, что и -v, но символы табуляции представляются как
- e то же, что и v но в конце строк ставится символ $
- s — подавляет вывод пустых строк.
- u — включает режим небуферизуемого вывода. В ряде случаев буферизация позволяет повысить производительность, но она же может вызывать проблемы при работе с некоторыми устройствами текстового вывода, например терминалами Впрочем, обычно эта опция не дает никакого эффекта

Как уже говорилось, команду cat часто применяют для просмотра содержимого файлов. Например, команда cat somefile.txt выводит на экран файл somefile.txt Это удобно, если размер файла невелик или команда вызывается в терминальном окне, где есть средства прокрутки В случае длинных файлов лучше пользоваться программами постраничной разбивки, в частности less (см главу 5).

При наличии нескольких аргументов команда cat отображает файлы последовательно. В сочетании с операторами переадресации это позволяет копировать несколько файлов в один. Рассмотрим пример:

```
$ cat file1.txt file2.txt > combined-file.txt
```

В полученный файл combined-file.txt будет записано содержимое файла file1.txt, а затем - содержимое файла file2.txt. Разрешается включать нумерацию строк, отображать непечатаемые символы и т.д.

Команда dd

Команду dd можно рассматривать как усовершенствованную версию команды cp. Она позволяет выполнять нестандартные преобразования данных, например копировать часть файла или переводить содержимое файла из нижнего регистра в верхний. В общем виде синтаксис команды таков:

```
dd [операнд[...]]
```

Основные возможности команды определяются ее операндами. Полный их перечень приведен на man-странице команды. Рассмотрим наиболее полезные операнды:

- **if=***файл* — задает входной файл. Если он не указан, команда считывает данные из стандартного входного потока (stdin).
- **of=***файл* — задает выходной файл. Если он не указан, команда записывает данные в стандартный выходной поток (stdout).
- **bs=***размер* — задает размер блока для операций чтения и записи. Это значение не играет особой роли, но оно упрощает вычисления при копировании части файла. Например, если известно, что размер копируемого фрагмента составляет 23 Кбайт, можно сделать следующие установки: bs=1024 и count=23.
- **count=***число блоков* — в сочетании с операндом bs определяет объем копируемых данных.
- **seek=***число блоков* — заставляет команду пропустить заданное число блоков выходного файла, прежде чем начинать записывать в него данные. Этот операнд можно использовать для перезаписи фрагмента файла.
- **skip=***число блоков* — заставляет команду пропустить заданное число блоков при чтении входного файла. Этот операнд можно использовать для копирования фрагмента файла.
- **conv=***преобразование* — заставляет команду преобразовать копируемые данные. Например, операнд conv=lcase задает перевод всех символов в нижний регистр, а операнд conv=ucase — в верхний регистр. Операнд conv=swab вызывает попарную перестановку байтов во входном потоке (это иногда необходимо при передаче бинарных файлов на компьютер с другим порядком следования байтов). О других преобразованиях рассказывается на man-странице команды.

Команду dd часто используют в сочетании с различными файлами устройств. Рассмотрим следующие примеры:

```
$ dd if=/dev/zero of=blank.img bs=1024 count 1440
$ dd if=floppy img of=/dev/fd0
```

Первая команда копирует 1440 Кбайт из устройства /dev/zero в файл blank.img. Специальное устройство /dev/zero реализуется ядром. При чтении из него всегда возвращается значение 0. Таким образом, эта команда формирует "пустой" файл объемом 1440 Кбайт, который может использоваться, к примеру, для создания дискового образа. Вторая команда осуществляет копирование в обратном направлении – из

файла floppy.img в устройство /dev/fd0 (дисковод гибких дисков). С помощью такой команды можно записывать образы дискет, например создавать загрузочные дискеты. Для этого, в принципе, подойдет команда cp и даже cat, но команда dd сообщает о том, успешно ли завершилась операция копирования, что особенно важно при работе с такими ненадежными носителями, как дискеты.

Удаление файлов

Для удаления файлов предназначена команда rm. Она может удалять не только отдельные файлы, но и целые иерархии каталогов. Синтаксис команды таков:

```
rm [-dfiPRrvW] файл[ ]
```

Ниже описаны опции команды:

- -d. Обычно команда не удаляет каталоги. Данная опция заставляет ее учитывать каталоги, но даже в этом случае каталог не будет удален, если в нем есть файлы или подкаталоги.

Если каталог выглядит пустым, но команда rm сообщает обратное, проверьте с помощью команды ls -a наличие скрытых файлов (их имена начинаются с точки).

- -f — заставляет команду удалять файлы, не выдавая запросов на подтверждение. Это удобно при удалении большого числа файлов, но опасно, если в аргументе *файл* присутствуют метасимволы.
- -i — заставляет команду выдавать запрос перед удалением файла.
- -P. Обычно, чтобы удалить файл, команда уничтожает его каталожную запись и модифицирует индексный дескриптор, указывая, что занимаемое дисковое пространство свободно. Сами данные при этом остаются на диске. Опция -P вызывает трехкратную перезапись данных перед изменением индексного дескриптора. Сначала поверх существующих данных записываются двоичные единицы, затем — двоичные нули и снова двоичные единицы. Это замедляет процесс удаления файла, зато значительно затрудняет низкоуровневое восстановление данных.
- -R или -r — заставляют команду рекурсивно удалить дерево каталогов. Наличие опции -d не требуется.
- -v — заставляет команду отображать имена удаляемых файлов.
- -W. При наличии этой опции команда попытается отменить удаление последнего файла. В настоящее время такая функция не поддерживается большинством файловых систем, но в будущем ситуация может измениться.

Последствия необдуманного применения команды rm могут оказаться весьма разрушительными, особенно если ее выполняет суперпользователь, указывая при этом опцию -R. Предположим, требуется удалить начальный каталог пользователя monica. Для этого должна быть введена команда rm -R /home/monica. Но если по ошибке вставить пробел между первой косой чертой и выражением home/monica, будут удалены все файлы компьютера! Во FreeBSD довольно слабые средства восстановления удаленных файлов, в сравнении с другими операционными системами, например Windows. Часто оказывается, что удаленный файл можно восстановить только с резервной копии.

Существует похожая команда `rmdir`, которая специально предназначена для удаления каталогов. Синтаксис ее вызова проще

```
rmdir [-p] каталог[, ]
```

Опция `-p` заставляет команду полностью удалить дерево каталогов, при условии что все каталоги пусты. Подобно `rm` команда `rmdir` не удалит каталог, в котором есть файлы. Для удаления иерархии каталогов проще пользоваться командой `rm -R`, но ее применение связано с определенным риском.

Ссылки на файлы

Ссылка (link) — это файловый указатель, позволяющий одному файлу иметь две и более каталожные записи. Аналогичные концепции существуют и в других операционных системах: ярлык (shortcut) в Windows, псевдоним (alias) в Mac OS и др. Во FreeBSD поддерживаются два типа файловых ссылок: *жесткие* и *символические* (рис. 8.1).

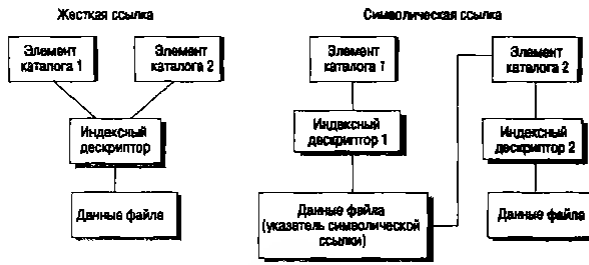


Рис. 8.1. Жесткая ссылка — это дополнительная каталожная запись для того же файла, а символическая ссылка — это специальный файл, хранящий имя другого файла

Жесткая ссылка представляет собой дополнительную каталожную запись, связанную с тем же самым файлом данных через тот же самый *индексный дескриптор* — структуру данных, сообщающую системе, где на диске хранится содержимое файла, когда был создан файл и т.д. Если удалить одну из каталожных записей (с помощью команды `rm`, например), индексный дескриптор и сам файл останутся неизменными и по-прежнему будут доступны. Доступ к файлу через любую из жестких ссылок осуществляется одинаково быстро, но тот факт, что альтернативная каталожная запись указывает непосредственно на искомый индексный дескриптор, означает, что жесткие ссылки должны находиться в одной файловой системе (т.е. в одном разделе или на одном съемном диске). Кроме того, некоторые программы резервного копирования, например `tar`, встречая жесткую ссылку, архивируют файл повторно (в случае символических ссылок этого не происходит). Во FreeBSD каталоги не могут быть жесткими ссылками.

Символическая ссылка — это файловый указатель более высокого уровня. Она хранит имя адресуемого файла, как если бы это были обычные данные. Во FreeBSD предусмотрена возможность получить доступ к целевому файлу по символической ссылке. Такой поиск занимает больше времени, чем в случае жесткой ссылки, хотя едва ли кто-то заметит разницу. Символические ссылки могут указывать на другие файловые системы, поэтому во многих случаях предпочитают работать именно с ни

ми Удаление ссылки не приводит к уничтожению файла, но если удалить или переместить сам файл, ссылка окажется *разорванной*, т.е. станет бесполезной до тех пор пока не будет создан новый файл, имя которого совпадет с содержимым ссылки Символическую ссылку можно безопасно перемещать, если в ней хранится полный путь к файлу В противном случае ссылка, скорее всего, будет разорвана

В стандартной инсталляции FreeBSD файловые ссылки используются в различных целях Например, некоторые файлы в каталоге `/etc` являются символическими ссылками на конфигурационные файлы, хранящиеся в других каталогах. Иногда ссылки приходится создавать в целях экономии дискового пространства Если, допустим, в корневой файловой системе не хватает места из-за того, что в каталог `/opt` были инсталлированы крупные программные пакеты, можно создать каталог `/usr/opt` (при условии, что каталог `/usr` находится в отдельном разделе), переместить в него содержимое каталога `/opt`, затем удалить исходный каталог `/opt` и воссоздать его в виде символической ссылки на каталог `/usr/opt`

Файловые ссылки создаются с помощью команды `ln`, синтаксис которой таков

```
n [-fhinsv] исходный_файл, .., целевой_файл | целевой_каталог]
```

Если в качестве целевого аргумента указывается файл, команда создает ссылку с этим именем Если целевым аргументом является существующий каталог, команда создает в нем ссылку с именем исходного файла. Некоторые опции команды, в частности `-f`, `-i` и `-v`, выполняют те же функции что и аналогичные опции комаид `cp` и `mv` Вот назначение остальных опций

- **h или -n** — запрещают команде интерпретировать целевой файл как символическую ссылку Используются совместно с опциями `-f` и `-s` для замены существующей ссылки на каталог другой ссылкой
- **s** По умолчанию команда создает жесткую ссылку Эта опция застааляет команду создавать символическую ссылку

В качестве примеров рассмотрим следующие команды

```
$ ln deeply/buried/directory/tree/file.txt ./
$ ln  s /mnt/zip/instructions.ps use-instructions.ps
# ln  s /opt/newprogram/bin/newprog /usr/local/bin
```

Первая команда создает в текущем каталоге жесткую ссылку, которая упрощает доступ к файлу, хранящемуся слишком глубоко в иерархии каталогов Поскольку в качестве целевого аргумента указано имя каталога (`./`), ссылка будет названа так же как и исходный файл (`file.txt`). Впоследствии можно удалить исходную каталожную запись, и файл останется доступен через новую ссылку

Во втором и третьем примерах создаются символические ссылки Первая из них делает файл, хранящийся в каталоге `/mnt/zip`, доступным в текущем каталоге под другим именем. Вторая делает исполняемый файл доступным под тем же именем в каталоге `/usr/local/bin`, который у большинства пользователей включен в переменную среды `PATH` (задает путь поиска программ) Эта последняя команда требует привилегий суперпользователя так как рядовые пользователи не имеют права записи в каталог `/usr/local/bin`

Поиск файлов

Во FreeBSD есть несколько команд для поиска файлов. Наиболее мощные из них — `find` и `grep` Они осуществляют поиск, соответственно, по *метаданным* (имя файла, время создания и т.д.) и по содержимому файлов.

Команда find

Команда find ищет файлы по атрибутам Базовый ее синтаксис таков

```
find [ ] путь_поиска[ ..] спецификация
```

На заметку Команда find поддерживает дополнительные опции, не описанные здесь. Дополнительная информация приведена на man-странице ко манды

Опция *-f путь_поиска* указывает команде, где необходимо начать поиск файлов Например, если известно, что файл находится в каталоге /usr/local, задайте опцию f /usr/local, чтобы ускорить поиск. Это очень полезный прием, поскольку иначе команда будет просматривать каждый файл в дереве каталогов FreeBSD.

Основные функциональные возможности команды скрыты в аргументе *спецификация*, состоящем из одной или нескольких поисковых опций Эти опции задают искомые атрибуты файлов и определяют особенности поиска Ниже перечислены наиболее важные опции

- **-atime** *время* — поиск фаилов, обращение к которым происходило в течение заданного времени Например, выражение -atime -12h означает, что доступ к файлу имел место в течение последних 12 часов. Значение времени задается в виде отрицательного числа с суффиксом s (секунды) m (минуты) h (часы) d (дни) или w (недели).
- **ctime** *время* - аналогична предыдущей опции но обозначает время созда ния файла
- **delete** - в сочетании с другими опциями заставляет команду удалить наи денные файлы Желательно сначала выполнить команду без этой опции чтобы просмотреть список удаляемых файлов
- **group** *группа* — поиск файлов, принадлежащих заданной группе В качестве аргумента может стоять как имя группы, так и ее идентификатор
- **-links** *число* - поиск файлов с заданным числом жестких ссылок
- **maxdepth** *глубина* — заставляет команду прекратить поиск при достижении заданной глубины дерева каталогов
- **mtime** *время* — аналогична опции at me но обозначает время модифика ции файла
- **name** *шаблон* — поиск файлов по имени В качестве шаблона может быть указано простое имя или строка с метасимволами, такими как '*' и '?'. Во это ром случае строку необходимо взять в кавычки, чтобы интерпретатор команд не раскрыл метасимволы.
- **newer** *имя_файла* — поиск файлов, созданных позже, чем заданный файл
- **perm [+|-]***режим* — поиск файяов с определенными правами доступа Аргу мент *режим* задается так же, как и в команде chmod Если отсутствует префикс '+' или '-', аргумент должен в точности соответствовать атрибутам файла. При наличии префикса '+' команда ищет файлы, у которых установлен хотя бы один из заданных битов доступа. В случае префикса '-' у файла должны быть установлены все указанные биты, а остальные биты игнорируются.
- **-size** *n*[c] — поиск файлов по размеру Если присутствует суффикс c раз мер задается в байтах, иначе в виде количества 512-байтовых блоков
- **type** *тип_файла* — поиск файлов заданного типа Коды типов перечислены в табл 8 1 Дополнительный код f обозначает обычный файл

- **user** *имя_пользователя* — поиск файлов принадлежащих заданному пользователю. В качестве аргумента может стоять как имя пользователя так и его идентификатор.

Вот несколько примеров

```
$ find ./ -name "*txt"
$ find /usr/local -perm +u+s
# find /etc -mtime  24h -name "*rc"
```

Первая команда находит в текущем каталоге вее файлы имена которых заканчиваются на txt. Вторая команда ищет в каталоге /usr/local файлы с установленным битом SUID Третья команда ищет в каталоге /etc все файлы, имена которых заканчиваются на rc и которые модифицировались на протяжении последних 24 часов. Эта команда обозначена как привилегированная, поскольку в каталоге /etc есть подкаталоги, недоступные рядовым пользователям Для последних такая команда выдаст неполные результаты

Комаида find отображает имена всех файлов, соответствующих условию поиска Если ни одного файла не найдено, ничего не выводится, поэтому иногда возникает впечатление, будто команда не выполнена

Команды grep и egrep

Команда find ищет файлы по имени, дате модификации, типу и т д., но иногда этого недостаточно. Другой распространенный способ поиска основан на анализе содержимого файлов. Здесь на помощь приходит команда grep, которая возвращает имена файлов, содержащих заданные строки. В случае текстовых файлов отображаются также строки, в которых найдено совпадение. Синтаксис команды таков:

```
grep [опции] [шаблон | e шаблон | -f файл] [файл[...]]
```

Существует похожая команда egrep отличающаяся от grep лишь синтаксисом шаблонов поиска. Рассмотрим наиболее важные опции команды grep.

- **-A** *число* — заставляет команду отображать указанное число строк после искомой
- **B** *число* — заставляет команду отображать указанное чисяо строк перед искомой
- **-c или --count** — застааляет комаияу сообщать число совпавших строк вместо того чтобы отображать сами строхи
- **-F** По умолчанию команда интерпретирует шаблон как регулярное выражение (об этом рассказывается ниже) Эта опция заставляет команду воспринимать шаблон буквально, что ускоряет поиск некоторых строк
- **-i или --ignore-case**. По умолчанию команда распознает регистр символов в шаблоне поиска Эта опция включает режим поиска без учета регистра
- **-r или - recursive** — включает режим рекурсивного поиска в дереве каталогов
- **v или -invert-match** — заставляет команду инвертировать смыся операции будут отображаться строки, *не соответствующие* шаблону

Шаблон поиска — это строка специального вида, перед которой может стоять опция e (она позволяет различать шаблоны, начинающиеся с дефиса, которые в противном случае будут интерпретированы как опции команды grep). Если определенные вияы поиска выполняются достаточно регулярно, поместите шаблоны поиска в файл и подключайте его с помощью опции -f *файл* Шаблон представляет собой *регулярное выражение*, т.е. спецификацию искомой строки, содержащую различные метасимволы. Регулярные выражения бывают довольно сложными, и здесь мы лишь в общих чертах рассмотрим, как они составляются Иногда регулярные выражения

нужно брать в кавычки особенно если они содержат символы * и ? чтобы интерпретатор команд не воспринял их как имена файлов и не раскрыл раньше времени. Вот основные особенности регулярных выражений команды grep

- **Литералы.** Большинство символов интерпретируется буквально, т.е. они соответствуют самим себе Например, строка trust всегда задает поиск слова trust
- **Наборы символов.** Строка заключенная в квадратные скобки например [aeiou] соответствует любому из перечисленных символов Если первым в списке стоит символ '^', то строка соответствует любому символу не входящему в список. Чтобы задать диапазон символов, укажите начальный символ, дефис и конечный символ. К примеру, выражению [0-9] соответствует любая цифра
- **Одиночные символы.** Точка ( ) соответствует любому одиночному символу, выражение \w — любому алфавитно-цифровому символу (цифра или буква), выражение \W — любому специальному символу (знак препинания непечатаемый символ и т.д.)
- **Коды повторения.** Существуют специальные коды, указывающие на то, что предыдущий элемент повторяется определенное число раз. В частности, выражение \? обозначает ни одного или рокио одно повторение, знак * — ни одиого ияи несколько повторений, выражение \+ – одно или более повторений В команде egrep обратная косая черта не нужна
- **Альтернативные варианты.** Вертикальная черта ( ) разделяет два регулярных выражения Искомая строка может соответствовать любому из них

Некоторые символы в составе регулярных выражений трактуются особым образом. Чтобы включить такой символ в шаблон поиска, нужно защитить его от интерпретации с помощью обратной косой черты. Например, выражению \? в шаблоне поиска соответствует знак вопроса Символ обратной косой черты при поиске тоже нужно защищать (\\). Можно также воспользоваться опцией F чтобы заставить команду grep интерпретировать весь шаблон буквально

По умолчанию команда grep принимает данные из ствндартного входного потока. Это позволяет легко подключать ее к каналам ввода-вывода Но если команда используется для поиска файлов, ей передается имя файла или каталога либо шаблон имени

Рассмотрим примеры

```
$ grep Byers *.tex
$ grep trust.*one gullible/* paranoid/*
$ grep "Chapter.*\.\+[0-9]" *.txt
$ egrep "Chapter.*\ +[0-9]" *.txt
$ grep -r "copyright 200[12]"  /
```

Первая команда ищет в текущем каталоге файлы с расширением tex содержащие строку Byers Это наиболее простой вариант команды. Во втором примере в каталогах gullible и paranoid ищутся файлы содержащие в одной строке слова trust и one Шаблону поиска соответствуют выражения trust everyone и trust no one. Обратите внимание на различную интерпретацию символа '*'. В шаблоне trust.*one он является частью регулярного выражения и указывает на возможность разделения двух слов, а в шаблонах gullible/* и paranoid/* это метасимвол заменяющий спецификацию имени файла Третья команда ищет файлы с расширением .txt, содержащие слово Chapter за которым могут следовать произвольные символы (.*), затем одна или несколько точек (\.\+) и цифра ([0-9]). С помощью такой команды можно находить файлы, содержащие оглавление книги. В них будут присутствовать строки примерно такого вида

```
Chapter 23 .. ..479
```

В четвертом примере делается то же самое, но используется команда `egrep`, поддерживающая расширенный синтаксис регулярных выражений. Это позволяет не указывать обратную косую черту перед оператором +. Наконец, пятая команда просматривает все дерево каталогов в поиске файлов, содержащих строку `copyright 2001` или `copyright 2002`. Обычно команда `grep` воспринимает последний аргумент как спецификацию имен файлов, но в случае рекурсивного поиска можно указывать список имен каталогов.

Когда команда `grep` находит совпадение в текстовом файле, она отображает имя файла и соответствующую строку (одну или несколько, если совпадений более одного). В случае же бинарного файла выдается лишь сообщение вида `binary file` *<имя файла>* `matches`.

Резервное копирование

Под резервным копированием понимается сохранение файлов вне компьютера, чтобы они были доступны в различных аварийных ситуациях, как-то: случайное удаление важного файла, уничтожение или кража компьютера и т.д. Как правило, резервные копии держат, что называется, под рукой, но для максимальной надежности рекомендуется хранить их в другом помещении или даже другом здании. Это послужит защитой на случай крупной катастрофы, такой как пожар или стихийное бедствие.

Внимание! Позаботьтесь о надежной защите резервного носителя от посторонних глаз. Файлы, недоступные рядовым пользователям системы, часто оказываются "беззащитными" на резервном носителе, поскольку распространенные устройства резервного копирования не поддерживают ограничений FreeBSD, связанных с владельцами и правами доступа.

Резервные носители

Чаще всего для резервного копирования используют магнитные ленты. Накопители на магнитных лентах стоят от ста до нескольких тысяч долларов, в зависимости от емкости носителя, скорости передачи данных и надежности. В рабочих станциях и небольших серверах применяются накопители стоимостью от 200 до 1500 долларов. Наиболее дешевыми устройствами являются накопители Travan, однако у них дорогостоящие ленты (около 40 долларов). У более дорогих устройств, например DAT (Digital Audio Tape — цифровая аудиолента), стоимость носителя может быть меньшей, но даже если она выше, это компенсируется многократным увеличением емкости, поэтому к оценке стоимости следует подходить комплексно.

Преимуществом магнитных лент является высокая емкость (современные ленты вмещают десятки гигабайт), довольно высокая скорость передачи данных и умеренная стоимость одного гигабайта архивируемых данных. Потенциальные конкуренты, например съемные диски, имеют гораздо меньшую емкость и гораздо более высокую удельную стоимость данных. Скажем, для архивирования современной системы потребуется не один десяток Zip-дисков емкостью 100 Мбайт, в то время как лент — всего одна-две.

Особого внимания заслуживают компакт-диски. Компакт-диск формата CD-R (с однократной записью) имеет емкость 650 Мбайт, что достаточно (особенно с учетом сжатия) для хранения базовой инсталляции FreeBSD. Такую резервную копию можно держать на случай аварийного восстановления всей системы, используя магнитные ленты для регулярного архивирования пользовательских данных. Компакт-диски очень дешевы и к тому же могут читаться на любом компьютере, поэтому в сетевой

среде их применяют для хранения резервных копий клиентских систем Это очень надежные носители с длительным сроком службы (по разным оценкам — от 20 до 100 лет) Более новые записываемые диски формата DVD (Digital Versatile Disk — универсальный цифровой диск) имеют емкость несколько гигабайт, но пока что они значительно дороже, чем компакт-диски формата CD-R да и устройства чтения таких дисков еще не получили всеобщего распространения

Утилиты резервного копирования

Во FreeBSD есть несколько программ резервного копирования Большинство из них ориентировано на работу с магнитными лентами, но их можно использовать и с другими носителями Вот наиболее популярные программы

- **tar** Утилита `tar` (tape archiver — архиватор лент) уже давно существует в UNIX и считается стандартной. Как указывает ее название, она создает архивы на магнитных лентах Эта утилита превратилась также в популярное средство подготовки дистрибутивных пакетов Утилита может создавать специальные архивные файлы, включающие в себя другие файлы что упрощает распространение программ
- **dump**. Утилита `dump`, как и `tar`, тоже считается стандартной Необычной ее особенностью является тесная связь со структурами данных файловой системы Это означает, что некоторые файловые системы, например разделы FAT, нельзя архивировать с ее помощью Кроме того, данные желательно восстанавливать в файловой системе того же типа. Если, к примеру, архив создавался во FreeBSD, то, возможно, не удастся восстановить его в Linux где используется файловая система `ext2`. В отличие от большинства программ резервного копирования для восстановления данных утилита `dump` использует отдельную программу, называемую `restore`
- **cpio** Программа `cpio` в целом эквивалентна утилите `tar` но имеет другой синтаксис и создает архивы иного формата
- **Утилиты для работы с компакт-дисками формата CD R**. Если в качестве резервного носителя выбран компакт-диск формата CD R нужно использовать специальные программные средства Процесс записи на диск разбивается на два этапа. Сначала с помощью программы `mkisofs` создается *образ* файловой системы ISO-9660, содержащей нужные файлы Затем посредством команды `cdrecord` файл образа переносится на пустой компакт-диск. С точки зрения резервного копирования нежелательно включать файлы непосредственно в файл образа, поскольку при этом теряется часть атрибутов, например сбрасываются биты записи Лучше с помощью утилиты `tar`, `dump` или `cpio` создать архив и записать на компакт-диск его, а не отдельные файлы

Все эти программы являются стандартными компонентами FreeBSD Ниже будет рассмотрена утилита `tar`, так как она чрезвычайно популярна и к тому же широко применяется не только для резервного копирования.

Утилита tar

В базовом виде синтаксис утилиты `tar` выглядит так

```
tar функция [модификатор_функции, ..]] [файл[ ..]]
```

Аргумент *функция* задает выполняемое действие, например `--extract` (или просто `e`) — извлечение файлов из архива Необязательный модификатор определяет параметры операции Наиболее важные функции и модификаторы описаны в табл 8.5 и 8.6. В конце командной строки приводится список файлов и каталогов

Таблица 8.5 Важные функции утилиты `tar`

Функция	*Обозначение*	*Назначение*
`--create`	`c`	Создание нового архива
`--concatenate` или `--catenate`	`A`	Добавление содержимого одного архива в другой
`--append`	`r`	Добавление новых файлов в существующий архив
`--update`	`u`	Добавление обновленных файлов в существующий архив (не работает в случае магнитных лент)
`--diff` или `--compare`	`d`	Сравнение содержимого архива с заданными файлами и отображение различий
`--list`	`t`	Вывод имен файлов, содержащихся в архиве
`--extract` или `--get`	`x`	Извлечение файлов из архива
`--delete`	—	Удаление файлов из архива (не работает в случае магнитных лент)

Таблица 8.6 Важные модификаторы функций утилиты `tar`

Модификатор	*Обозначение*	*Назначение*
`--file [`*`узел:`*`]`*`файл`*	`f`	Использование указанного файла в качестве архивного. Если задан узел, осуществляется доступ к удаленному компьютеру
`--listed-incremental` *`файл`*	`g`	Инкрементное резервное копирование, при котором архивируются только те файлы, которые были модифицированы с момента последнего архивирования. Указанный файл содержит данные о ранее архивировавшихся файлах
`--bunzip2`, `--bunzip` или `--bz`	`y` или `j`	Сжатие архива с помощью утилиты `bzip2`
`--one-file-system`	`l`	Обработка файлов только в одной файловой системе (в одном разделе)
`--tape-length` *`N`*	`L`	Создание архива размером *N* Кбайт, после чего делается пауза, чтобы оператор мог сменить ленту
`--multi-volume`	`M`	Обработка многотомного архива
`--same-permissions` или `--preserve-permissions`	`p`	Обработка всей информации о правах доступа
`--absolute-paths`	`P`	Добавление начальной косой черты к полным именам файлов
`--verbose`	`v`	Вывод списка файлов, обрабатываемых в режиме `--create` или `--extract`, и отображение более полной информации о правах доступа в режиме `--list`
`--exclude` *`шаблон`*	—	Игнорирование файлов, соответствующих шаблону
`--gzip` или `--gunzip`	`z`	Сжатие архива с помощью утилиты `gzip`

Как правило, один спецификатор функции используется совместно с одним или несколькими модификаторами. Например, следующая команда создает резервную копию разделов `/home` и `/var` на ленточном SCSI-устройстве

```
# tar --create --verbose --file /dev/sa0 /home /var
```

Эту же команду можно записать короче

```
# tar cvf /dev/sa0 /home /var
```

Эти команды иллюстрируют принцип, общий для утилиты tar и других программ резервного копирования: ленточное устройство трактуется как обычный файл. В частности, файлу /dev/sa0 соответствует SCSI-накопитель на магнитных лентах, а файлу /dev/ast0 — EIDE-накопитель. Если к системе подключено несколько однотипных накопителей, они различаются по номерам. Например, файлу /dev/sa1 соответствует второй SCSI-накопитель на магнитных лентах.

При первом обращении к магнитной ленте может появиться сообщение об ошибке. Введите **mt fsf 1**, извлеките ленту и подключите заново. Утилита mt применяется и в других операциях с магнитными лентами, например для включения встроенных функций сжатия накопителей.

Утилита tar сначала создает архив, а потом сжимает его. Если на ленте есть дефект, все данные, хранящиеся после дефектного участка, окажутся потерянными. По этой причине не рекомендуется использовать опции -bzip и --gzip при создании архивов на магнитных лентах. Лучше включить встроенный режим сжатия ленточного устройства. Жесткие диски и компакт-диски более надежны, чем ленты, поэтому сжатие архивов при работе с ними обычно не вызывает проблем.

Модификатор --one-file-system играет важную роль, не позволяя утилите архивировать виртуальные файловые системы, такие как /dev и /proc. Их содержимое динамически генерируется ядром FreeBSD. На ленте эти файловые системы будут лишь напрасно занимать место, а вот попытка восстановить их может вызвать серьезные проблемы. Если используется модификатор --one-file-system, нужно в отдельном файле указать список всех архивируемых разделов. Предположим, в системе имеются разделы /, /tmp, /var, /usr и /home. Такая система архивируется следующей командой:

```
# tar cvplf /dev/sa0 /home /var / /usr /tmp
```

В этом примере первыми перечислены те разделы, которые чаще приходится восстанавливать (чем ближе к началу ленты находятся файлы, тем быстрее они восстанавливаются). Раздел /tmp обычно не архивируют, поскольку его содержимое, как правило, уничтожается при перезагрузке.

По умолчанию команда tar удаляет начальную косую черту из полных имен файлов во избежание проблем при восстановлении. Предположим, возникла аварийная ситуация и дерево каталогов FreeBSD монтируется в другой системе к нестандартному каталогу, например /mnt. Тогда нужно перейти в каталог /mnt и воспользоваться опцией --extract, чтобы восстановить данные, не повреждая файлы резервной системы. При наличии модификатора --absolute-paths утилита tar сохраняет начальную косую черту в именах файлов. Это может понадобиться для архивирования файлов, хранящихся в строго определенных каталогах, но даже в этом случае лучше создавать стандартные архивы.

Прежде чем разархивировать tar-файл, проверьте его содержимое, воспользовавшись опцией --list. Злоумышленник может создать архив в режиме --absolute-paths, включив в него файлы, которые при разархивировании перезапишут стандартные системные утилиты, что позволит взломать систему или нарушить ее работу.

Восстановление данных

С резервным копированием данных связано и их восстановление. Утилиты `tar` и `cpio` делают это самостоятельно. Их проще всего применять, если система находится в рабочем состоянии и нужно восстановить лишь несколько файлов. Предположим пользователь `walter` приходит к администратору и заявляет что случайно удалил важный файл `report78.tex` из своего начального каталога Вот как восстановить этот файл

```
# cd /
# tar xvpf /dev/sa0 home/walter/report78.tex
```

На заметку Поскольку утилита `tar` в процессе записи файла в архив удаляет начальную косую черту из его имени нужно перейти в корневой каталог или в тот каталог, где была создана резервная копия, и ввести имя файла без начальной косой черты

Если пользователь не помнит точное имя файла вызовите утилиту `tar` с опцией `list`, чтобы найти нужный файл Можно также восстановить все дерево каталогов Если не указать имя файла, утилита `tar` восстановит все содержимое резервной копии хотя это не всегда требуется

Совет Попрактикуйтесь в создании резервных копий на временном каталоге Скопируйте в него несколько файлов, а затем восстановите их в другом каталоге и проверьте, все ли прошло успешно

В экстренных случаях приходится восстанавливать всю систему или по крайней мере минимальный набор файлов, обеспечивающих ее функционирование Это может быть вызвано, к примеру, сбоем жесткого диска или взломом системы с последующим повреждением ее файлов. Есть несколько способов решения такой проблемы.

- **Создание резервной системы** Как описывается в главе 32, можно воспользоваться аварийной загрузочной системой, чтобы загрузить FreeBSD, подготовить новые разделы и восстановить все данные. Эту процедуру нужно запланировать *до того*, как произойдет катастрофа
- **Частичная переинсталляция**. Можно инсталлировать FreeBSD в минимальном варианте, к примеру на старый жесткий диск, и использовать ее в качестве аварийной системы для восстановления данных на более крупном диске После этого следует переконфигурировать диски, чтобы диск с восстановленной системой стал загрузочным
- **Полная переинсталляция** Можно переинсталлировать FreeBSD нормальным способом, а затем воспользоваться резервной копией для восстановления только ключевых каталогов, которые наверняка изменились, например `/home`. Такой подход становится тем более затруднительным, чем сильнее система отличается от стандартного варианта инсталляции Даже если все локальные настройки четко зафиксированы, на восстановление прежнего состояния системы уйдет не один час

Полная переинсталляция системы может обернуться настоящим кошмаром, особенно если вы не подготовлены к этому. Продумайте заранее свои действия на случай выхода из строя жесткого диска или другой аварии. Как вы будете восстанавливать данные? Разработайте план восстановления и по возможности попрактикуйтесь в его применении Попробуйте, к примеру, восстановить систему на резервном компьютере и загрузиться с него (при условии, что аппаратные различия несущественны) Чем более важную роль играет система, тем больше времени нужно потратить на защиту от неприятностей

Резюме

Во FreeBSD с файлами связаны различные метаданные, включая идентификатор владельца, идентификатор группы и права доступа. Эта информация важна для выполнения определенных операций над файлами. Например, невозможно скопировать файл, если его нельзя прочесть, а если файл копируется или перемещается в другой каталог, то нужно иметь право записи в этот каталог. Во FreeBSD есть целый ряд команд манипулирования файлами, позволяющих модифицировать метаданные, перемещать, копировать, удалять, искать файлы и т.д. Эти команды чрезвычайно полезны как для рядовых пользователей, так и для администраторов FreeBSD. Файлы нужно также защищать от возможного повреждения, создавая их резервные копии на внешних носителях, таких как компакт-диски и магнитные ленты. Наличие такой копии позволит быстро восстановить данные в случае аварии или сбоя.

Полный справочник по

Глава 9

Конфигурирование принтера

Несмотря на бурное развитие компьютерных технологий, безбумажный офис — по прежнему несбыточная мечта для большинства организаций, и средства печати до сих пор не утратили своей важности Модель печати во FreeBSD покажется достаточно непривычной для пользователей Windows или Mac OS поэтому мы нач нем эту главу с рассмотрения того как вообще организуется печать во FreeBSD. Далее будет описано решение практических задач, таких как создание очереди на печать конфигурирование программы Ghostscript (играет роль драйвера принтера) управле ние очередью и сетевым доступом к принтеру.

Модель печати во FreeBSD

В большинстве современных операционных систем используется *очередь на печать* которая представляет собой совокупность документов, посылаемых на принтер Во FreeBSD существует множество программ, позволяющих ставить документы в очередь, определять приоритеты и выполнять другие манипуляции с *заданиями* (так называют документы, находящиеся в очереди), приводить задания к виду, понятному принтеру, и посылать задания на выполнение Чтобы правильно сконфигурировать очередь на печать, важно понимать, как все эти программы взаимодействуют друг с другом

Очередь на печать

Вспомним, как принтеры работали в DOS. Программы направляли данные непо средственно в порт принтера Операционная система практически не контролировала доступ к принтеру, а за представление данных в формате, понятном принтеру, отвечали сами программы. С появлением многозадачных операционных систем возникла необходимость в более сложной модели печати В системе наподобие FreeBSD программам не разрешен прямой доступ к портам принтеров Представим себе две программы, пытающиеся записать данные в порт одновременно В лучшем случае одной из программ будет отказано в доступе, и соответствующее задание не напечатается Пользователям придется сражаться друг с другом в попытке направить на печать задание в тот короткий промежуток времени, когда порт освобождается. В худшем случае данные обеих программ начнут обрабатываться вперемежку, и результаты такой печати останется отправить в мусорную корзину

Решением проблемы является очередь на печать, которая позволяет операционной системе принимать многочисленные задания и направлять их на принтер по порядку Во FreeBSD, когда одно задание посылается на принтер, остальные хранятся на диске в специальном буфере, называемом *областью спулинга* (как правило, она находится в каталоге `/var/spool/lpd`) Операционная система способна обрабатывать произвольное количество заданий печати На практике их число, конечно же, ограничено свободным дисковым пространством в разделе `/var` или в том разделе, где находится каталог `/var/spool/lpd`

Очередь на печать обеспечивает, прямо или косвенно следующие преимущества

- **Планирование заданий печати** По умолчанию задания обрабатываются по принципу "первый поступил — первым обслужен" Это устраняет проблемы возникающие, когда несколько пользователей или программ пытаются получить доступ к одному и тому же ограниченному ресурсу. Приложение может поставив в очередь большое задание, завершить работу, но задание все равно будет напечатано Разрешается также менять порядок заданий в очереди

- **Управление принтером** Очередей на печать может быть несколько и каждой из них присваивается имя (должно быть легко запоминаемым). За принтером разрешается закреплять несколько имен, и можно сконфигурировать его так, чтобы он функционировал по-разному в зависимости от того, по какому имени к нему обращаются. При замене или обновлении принтера его можно связать со старым именем что избавит пользователей от лишних трудностей
- **Фильтрация заданий** Одна из наиболее удачных особенностей очередей на печать во FreeBSD — возможность применения фильтров *Фильтр* — это программа, которая обрабатывает задание до того, как оно будет передано на принтер. Чаще всего такие фильтры вызывают программу Ghostscript (описывается далее).
- **Сетевой доступ к принтеру** Можно легко сконфигурировать систему на передачу заданий принтеру, подключенному к другому серверу печати и принимать задания на печать от других компьютеров. (Конфигурация будет простой, если в других системах применяется BSD-модель печати Организовать взаимодействие с клиентами и серверами Windows или Mac OS гораздо сложнее.) Эта тема тоже рассматривается в настоящей главе

Во FreeBSD используется система печати *BSD LPD*, названная так по имени операционной системы, в которой она появилась (BSD), и основного компонента — *демона построчной печати* (Line Printer Daemon, LPD) Существуют и другие, более современные системы Наиболее значительная из них — *LPRng*, совместимая со своей предшественницей Большая часть информации, приведенной в этой главе, относится к LPRng и BSD LPD. Другая перспективная система, CUPS (Common UNIX Printing System — общая система печати в UNIX), отличается от BSD LPD гораздо больше В ней поддерживаются расширенные возможности, например передача приложению сведений об особенностях принтера. Следствием этого является несовместимость процедур конфигурирования, хотя в CUPS используются аналогичные программы для выдачи заданий на печать, поэтому с точки зрения пользователей системы CUPS и BSD LPD работают схожим образом Как LPRng так и CUPS включены в стандартный дистрибутив FreeBSD

Постановка заданий в очередь

Если система сконфигурирована на использование принтера, то первое, что делает пользователь при работе с ним, выдает задание на печать. В BSD LPD это осуществляется с помощью команды `lpr` Ее можно вызывать непосредственно (чтобы распечатать файл, находящийся в одном из пользовательских каталогов) либо через другую программу (к примеру, текстовый процессор может поставить задание в очередь, передав созданный файл команде `lpr`).

Команда `lpr` принимает входной файл и передает его следующей программе подсистемы печати· демону `lpd` Эта команда является удобным посредником позволяя записывать файл в выбранную очередь

На заметку В LPRng и CUPS тоже есть команда `lpr`, которая работает почти так же, как и ее аналог в BSD LPD. А вот демоны печати у этих систем свои

Помните, что задание это файл. Даже если он не создается в явном виде, команда `lpr` и демон `lpd` трактуют задание именно как файл В процессе передачи на принтер задание в определенный момент обязательно оказывается на жестком диске в виде файла

Демон построчной печати

Основную часть работы, связанную с печатью файлов, выполняет демон `lpd`, который и дал название системе печати BSD LPD. Как и все демоны, он работает в фоновом режиме, ожидая поступления данных. Особенностью FreeBSD является то, что демон принимает задания, используя сетевые протоколы. Если печать ведется локально, команда `lpr` передает данные демону через интерфейс `localhost`. Это означает, что задание обрабатывается так, будто поступило по сети, хотя в действительности оно не покидало компьютер. Демон `lpd` способен принимать запросы и от других компьютеров, если его надлежащим образом сконфигурировать.

Когда поступает задание, демон `lpd` сохраняет его в каталоге спулинга указанного принтера и фиксирует для себя, в какой позиции очереди оно находится. Обработав предыдущие задания, демон переходит к следующему. Оно пропускается через фильтры, связанные с очередью, а затем записывается в порт принтера или отправляется на другой компьютер. После того как завершающий байт задания передан по назначению, демон `lpd` удаляет задание из очереди.

Язык PostScript

Важной особенностью очереди на печать является то, что она дает приложениям определенную независимость от принтера. В Windows, OS/2 или Mac OS для используемой модели принтера инсталлируется драйвер. Приложения осуществляют печать с помощью системно-зависимых команд, а операционная система преобразует их в команды принтера. При этом система может сообщить приложению функциональные особенности принтера, например максимальный размер страницы. Во FreeBSD все происходит немного иначе. Традиционно UNIX-системы устанавливались на мощных компьютерах, к которым подключались высокоскоростные, производительные *принтеры построчной печати*, не поддерживающие сложное форматирование, либо лазерные принтеры PostScript. Таким образом, в UNIX и FreeBSD программы печати выводят задания либо в текстовом виде, либо в формате PostScript, предполагая, что печать осуществляется на принтере одного из двух типов.

Драйверам принтеров в Windows никогда не достичь той степени унификации, которая свойственна UNIX-программам. Приложение генерирует файл формата PostScript, и этого достаточно. Недостатком такого подхода является то, что UNIX-программы знают о принтере гораздо меньше, чем их Windows-аналоги. К примеру, программа не может определить, поддерживает ли принтер цветную печать и какова его разрешающая способность. Соответствующие установки закладываются в программу либо вводятся пользователем в диалоговом окне при подготовке к печати. Обойти эти ограничения — одна из целей системы CUPS. Она позволяет связать с каждой очередью файл формата *PPD* (PostScript Printer Description — описание принтера PostScript) и приложение может запросить у CUPS информацию, хранящуюся в этом файле.

Совет Если нужно обеспечить доступ к возможностям принтера, которые не поддерживаются напрямую в языке PostScript, создайте несколько очередей на печать. Например, для печати на струйном принтере с разной разрешающей способностью могут понадобиться две или три очереди. Такие конфигурации будут рассматриваться в этой главе.

На настольных компьютерах печать в текстовом режиме не вызывает проблем, в отличие от PostScript-заданий, поскольку поддержка языка PostScript приводит к существенному увеличению стоимости лазерных и струйных принтеров нижнего класса. Решением проблемы является использование фильтра, который закрепляется за оче

редью печати и преобразует PostScript данные в формат, понятный принтеру. Собственно преобразование осуществляет программа Ghostscript (`http://www.cs.wisc.edu/~ghost`). Она является интерпретатором языка PostScript, работающим на компьютере, а не на принтере. В следующем разделе будет показано, как создать очередь на печать *без* фильтра Ghostscript, если принтер поддерживает язык PostScript или нужно печатать только текстовые данные с использованием одного стандартного шрифта. Подключение программы Ghostscript рассматривается в разделе "Конфигурирование программы Ghostscript".

Лишь немногие приложения FreeBSD, например GIMP, могут генерировать данные для конкретных принтеров. Полученные результаты не всегда оказываются лучше, чем при использовании языка PostScript и обычной очереди на печать. Попробуйте оба способа, чтобы определить наилучший из них. Только помните, что некоторые фильтры неправильно распознают данные, отформатированные с учетом особенностей принтера и отклоняют либо искажают их. Возможно для печати таких данных придется создать специальную очередь без фильтра.

При покупке нового принтера лучше выбрать модель с поддержкой PostScript. Некоторые PostScript-принтеры оснащаются интерпретаторами языка, лицензированными компанией Adobe, в остальных используются клоны этих интерпретаторов. И те, и другие работают одинаково хорошо. Есть принтеры, рекламируемые как PostScript-модели, хотя на самом деле в них используются программы типа Ghostscript, предназначенные для Windows. Такие модели ничуть не лучше, чем принтеры без поддержки PostScript, предназначенные для FreeBSD, поэтому убедитесь в том, что средства поддержки встроены в принтер.

Что касается аппаратного интерфейса принтера, то во FreeBSD лучше всего поддерживаются параллельные порты. USB-принтеры тоже работают достаточно хорошо, хотя не все USB-устройства корректно распознаются. Принтеры с последовательным интерфейсом RS-232 работают медленно: их лучше избегать. Некоторые модели верхнего класса оснащаются интерфейсом Ethernet. Если они поддерживают протоколы BSD LPD, то их можно использовать не только как принтеры, но и как серверы печати.

Создание очереди на печать

Чтобы организовать очередь на печать, нужно создать каталог спулинга и должным образом отредактировать файл `/etc/printcap`, в котором программы печати ищут определения принтеров. Но предварительно следует убедиться в работоспособности самого устройства; это позволит избежать бессмысленной отладки в случае возникновения проблем. В этом разделе рассматривается также применение простейшего фильтра печати.

Файлы устройств печати

Современные принтеры в основном подключаются к параллельным или USB-портам. Если рабочая станция сконфигурирована на работу с сетевым принтером, то большую часть этого раздела можно пропустить.

Во FreeBSD для доступа к параллельному порту принтера предназначен файл `/dev/lpt0`. При наличии нескольких параллельных портов второму из них будет сопоставлен файл `/dev/lpt1` и т.д. В случае USB-принтеров эквивалентные файлы устройств называются `/dev/ulpt0`, `/dev/ulpt1` и т.д. Доступ к последовательным портам RS-232 осуществляется через файлы `/dev/ttyd0`, `/dev/ttyd1` и т.д. Тестирование и оптимизация этих портов являются важными этапами конфигурирования принтера.

Тестирование файлов устройств

В первую очередь следует протестировать порт принтера Один из способов сделать это — скопировать небольшой файл в порт например

```
# cat /etc/shells > /dev/lpt0
```

На заметку Эту команду следует выполнять в режиме суперпользователя По умолчанию рядовые пользователи не имеют доступа к порту принтера на низком уровне

Некоторые лазерные PostScript-принтеры отказываются печатать обычные текстовые файлы, но можно вывести на печать PostScript-файл Если в системе инсталлирована программа Ghostscript, образцы таких файлов содержатся в ее каталоге `examples` Попробуйте следующую команду

```
# cat /usr/local/share/ghostscript/6 52/examples/escher ps
```

На заметку Не все PostScript-файлы генерируют выходные данные Впрочем к файлу `escher.ps`, поставляемому с программой Ghostscript, это не относится Он должен работать с любым PostScript-принтером. Кроме того, этот файл удобно использовать для проверки того может ли принтер воспроизводить цвета

Если принтер ничего не напечатал причин может быть несколько

- **Не поддерживается текстовый режим** Некоторые принтеры не могут печатать в текстовом режиме Их относят к семейству *WinPrinters*, или к *GDI-принтерам* (по названию интерфейса GDI в Windows) Такие принтеры плохо работают во FreeBSD Чтобы получить информацию о принтере и узнать, совместим ли он с программой Ghostscript, посетите Web узел `www linuxprinting.org` (несмотря на название узла, приводимые там сведения в равной степени относятся как к Linux, так и к FreeBSD) Если принтер отнесен к категории "paperweight", его лучше заменить
- **Не распознается формат** Некоторые принтеры отказываются принимать данные, если они не представлены в определенном формате. К этой категории относится ряд PostScript-принтеров Есть принтеры, которые требуют наличия специальных префиксных кодов, задающих формат данных Такие принтеры могут работать во FreeBSD, но простой тест не позволяет это проверить Посмотрите, загораются ли индикаторные лампочки при записи данных в порт принтера Если это происходит, продолжайте конфигурирование принтера В противном случае проверьте другие предположения
- **Отсутствуют драйверы** Если вы решили перекомпилировать ядро как описано в главе 12, не забудьте включить драйверы принтера, иначе на принтер нельзя будет посылать данные Во FreeBSD 5 0 такую ситуацию легко обнаружить по отсутствию соответствующих файлов устройств в каталоге `/dev` Перекомпилируйте ядро еще раз или загрузите прежнее Некоторые USB-принтеры некорректно распознаются системой, и для них нужно искать "заплату" к ядру
- **Неправильный режим параллельного порта** Не все режимы оптимизации параллельного порта можно использовать с принтерами. Поэкспериментируйте с различными установками, чтобы выяснить, какие из них работают
- **Аппаратная проблема** В отдельную категорию вынесена целая группа проблем дефектное оборудование, отключенные кабели, отсутствие питание на принтере и т д. Все это не имеет отношения к FreeBSD, поэтому за советом нужно обращаться к руководству пользователя

Оптимизация параллельного порта

Убедившись в том что порт принтера работает, можно приступать к оптимизации его производительности. Это касается параллельных портов, но не USB или последовательных портов RS-232. Для настройки производительности порта предназначена команда `lptcontrol`. Синтаксис ее вызова таков:

```
lptcontrol -i | -p    e | s [-d устройство]
```

Опция `-i` задает режим управления по прерываниям а опция `p` — режим опроса Первый из них устанавливается по умолчанию Для большинства устройств он обеспечивает более высокое быстродействие, но в то же время принтер резервирует за собой один из запросов на прерывание, а это ограниченные аппаратные ресурсы, которые могут понадобиться для других целей. Кроме того, есть сведения, что некоторые принтеры некорректно работают в этом режиме, поэтому иногда его нужно отключать.

Опции `-e` и `s` включают и отключают расширенные режимы, поддерживаемые портом приитера. Все материнские платы, выпускаемые с середины 90-х годов, поддерживают по крайней мере один такой режим, например EPP (Enhanced Parallel Port — улучшенный параллельный порт) или ECP (Extended Capability Port — порт с расширенными возможностями). В большинстве случаев включение такого режима приводит к повышению производительности Обратитесь к документации на материнскую плату или проверьте установки BIOS чтобы выяснить какие расширенные режимы доступны

Аргумент *устройство* задает файл порта принтера например `/dev/lpt0` (это устройство выбирается по умолчанию)

Обычно работу параллельного порта оптимизируют, когда очередь печати уже создана и работает Проверьте различные варианты настроек, чтобы выяснить какие из них дают наилучшие результаты Если окажется, что стандартные установки не оптимальны, добавьте соответствующие команды в сценарии запуска системы

Замечания по поводу сетевых принтеров

Если FreeBSD конфигурируется для работы с сетевым принтером, то низкоуровневую проверку выполнить не удастся Но необходимо проверить работает ли сеть (об этом рассказывается в главе 14) Попробуйте, если возможно, напечатать что-нибудь непосредственно с сервера печати и с другого клиентского компьютера. В некоторых случаях приходится отлаживать как клиентскую, так и серверную системы На с. 234 будет рассказано, как превратить систему в сервер печати

Создание каталога спулинга

Системе нужно где то хранить задания, выводимые на печать поэтому первый шаг в организации очереди на печать заключается в создании буферного каталога. Он называется *каталогом спулинга* и обычно располагается в каталоге `/var/spool/lpd`. Каталог спулинга принадлежит пользователю `daemon` и одноименной группе. Только они имеют доступ к нему для чтения и записи

Учетная запись `daemon` и группа `daemon` используются некоторыми серверамя FreeBSD для доступа к определенным файлам и каталогам что позволяет этим серверам отказаться от привилегий суперпользователя

Каталог спулинга создается следующим образом

```
# mkdir /var/spool/lpd/hp4000
# chown daemon:daemon /var/spool/lpd/hp4000
# chmod 770 /var/spool/lpd/hp4000
```

В данном примере каталог называется hp4000. Желательно присваивать каталогам имена, которые легко ассоциируются с названием соответствующей очереди на печать. В большинстве случаев просто используют одинаковые имена, поэтому созданный нами каталог будет связан с очередью hp4000.

Формат файла /etc/printcap

В конфигурировании принтера важную роль играет настройка файла /etc/printcap. Используя информацию, хранящуюся в этом файле, демон lpd узнает, как обрабатывать файлы, посылаемые на заданный принтер: какие фильтры применять, нужно ли печатать страницу-разделитель между заданиями, с каким портом или сетевым сервером работать и т.д. Некоторые программы читают из этого файла список доступных принтеров.

На заметку В системах BSD LPD и LPRng используется файл /etc/printcap примерно одинакового формата. Для CUPS он не нужен, но от него зависит работа некоторых программ, поэтому в случае CUPS обычно создается усеченная версия файла.

Файл /etc/printcap поддерживает большое число опций. Чтобы продемонстрировать их использование, рассмотрим пример описания принтера (листинг 9.1). Если система имеет доступ к нескольким принтерам или принтер может работать в различных режимах, таких записей будет больше одной.

Листинг 9.1. Пример записи в файле /etc/printcap

```
lp hp4000:\
        :lp=/dev/lpt0:\
        :br#115200:\
         sd=/var/spool/lpd/hp4000
        :mx#0 \
        :sh.
```

Каждая запись начинается с указания имен принтера. Большинство приложений работает со стандартным именем lp, которое в данном случае приведено первым (если система взаимодействует с несколькими принтерами, имя lp должно быть только у одного из них). Имена отделяются друг от друга вертикальной чертой (|).

В соответствии с форматом файла /etc/printcap каждое определение должно занимать одну строку, поля которой разделяются двоеточием. Но для повышения наглядности поля записывают в разных строках, используя обратную косую черту в качестве символа продолжения. В листинге 9.1 каждая строка, кроме последней, завершается таким символом. Вторая и последующие строки каждой записи даны с отступом, благодаря чему видно, где заканчивается одно определение и начинается другое.

Большинство опций — это двухсимвольные коды. Значение опции указывается после знака равенства или решетки (#). Все опции описаны на man-странице файла printcap. Рассмотрим наиболее распространенные из них.

- lp — задает файл порта принтера, например /dev/lpt0 или /dev/ulpt1. В случае сетевого принтера эту опцию следует опустить (имеется в виду принтер, подключенный к другому компьютеру или имеющий прямое сетевое соединение с системой; при совместном использовании принтера, подключенного к параллельному, последовательному или USB-порту, опция нужна).
- br — задает скорость передачи в бодах для принтеров с интерфейсом RS-232. В случае листинга 9.1 опция игнорируется, так как принтер имеет параллель-

ный интерфейс Обычно скорость передачи задают максимально высокой В большинстве случаев значения 115200 вполне достаточно

- **rm** — используется только с сетевыми принтерами и задает имя сервера печати, например `rm=gutenberg.threeroomco.com`. Если конфигурируется принтер, подключенный к локальному порту, эта опция должна отсутствовать.
- **rp** — используется совместно с опцией `rm` и задает название очереди на печать в удаленной системе. Например, если в удаленной системе принтер называется `wgprint`, то установка будет выглядеть так: `rp=wgprint` Желательно, чтобы очереди на печать на одном и том же принтере в разных системах назывались одинаково Как и в случае опции `rm`, при конфигурировании локального принтера опция `rp` должна отсутствовать
- **sd** — задает каталог спулинга Этот каталог должен существовать По умолчанию используется каталог `/var/spool/lpd`
- **mx** — задает предельный размер файлов, обрабатываемых демоном `lpd` Это позволяет предотвратить злоупотребления со стороны пользователей, имеющих доступ к принтеру. Значение 0 означает отсутствие ограничения Учтите, что размер файла не связан с числом печатаемых страниц. Ограничение необходимо чтобы пользователи не могли преднамеренно переполнить раздел `/var`.
- **sh** По умолчанию система BSD LPD вставляет между заданиями *страницу разделитель* чтобы пользователи могли различать, где чье задание. Это хорошая конфигурация для интенсивно эксплуатируемых серверов печати, но не для персональных принтеров Опция `sh` подавляет вывод страниц разделителей.
- **if** — задает имя *входного фильтра*, который можно использовать для различной обработки печатаемых файлов, например с целью преобразования текста в формат PostScript или PostScript-данных в формат, понимаемый конкретным принтером В листинге 9.1 эта опция отсутствует, поэтому задания печатаются без изменений. Такая конфигурация подходит для PostScript-принтеров или для случая, когда на обычном принтере печатаются только текстовые данные В остальных случаях входной фильтр необходим.

В целях тестирования можно создать файл `/etc/printcap` в котором отсутствует входной фильтр. Затем нужно запустить демон `lpd`, если это еще не сделано. Введите **`ps ax | grep lpd`**, чтобы узнать работает ли демон. Сам демон запускается командой **`lpd`**. Если он работал до того, как были внесены изменения в конфигурацию, введите **`killall lpd; lpd`** для уничтожения и перезапуска демона. Чтобы демон вызывался в процессе загрузки системы, нужно модифицировать сценарии запуска. Это можно сделать в раздаже `Startup` меню `Configuration` программы `sysinstall` (рис 9 1) Другой способ — добавить показанную ниже строку в файл `/etc/rc.conf` или поменять `NO` на `YES` если строка уже присутствует

```
lpd_enable="YES"
```

Убедившись в наличии демона, протестируйте его с помощью того же самого файла, что и раньше. Для этого применяется такая команда:

```
$ lpr -Php4000 /etc/shells
```

Она выводит файл `/etc/shells` на принтер `hp4000` Результат должен быть таким же как и в случае прямой записи файла в порт принтера. Если файл не печатается, проверьте все опции в файле `/etc/printcap` и удалите лишние Например, опции `rm`, `rp` и `if` будут вызывать проблемы, если они присутствуют но не проинициализированы Попробуйте напечатать что-нибудь в режиме суперпользователя, а затем — от имени рядового пользователя В обоих случаях результат должен быть одинаков Если

осуществляется печать на сетевой принтер, настройте конфигурацию сервера печати чтобы он принимал файлы от клиентской системы О том, как это сделать, рассказы вается на с. 234

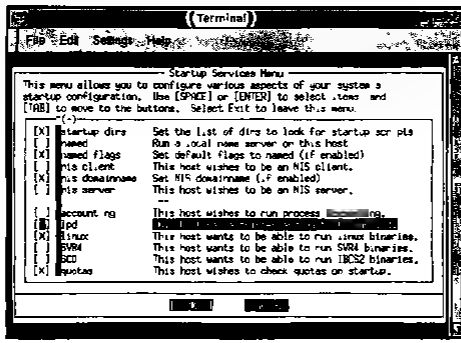


Рис 9 1 С помощью программы sysinstall можно заставить систему запускать демон lpd на этапе начальной загрузки

Использование простейшего фильтра

Столь мощной систему BSD LPD делает опция `if`, задающая входной фильтр В следующем разделе будет описано применение сложного фильтра совместно с программой Ghostscript для обработки различных типов файлов Но во многих случаях вполне достаточно более простого фильтра. Распространенный пример — преобразование текстового файла в формат PostScript для печати на PostScript-принтерах, которые не умеют анализировать текстовые данные Сначала нужно написать сценарий, который выступит в качестве фильтра Образец такого сценария показан в листинге 9 2 Сценарий читает входной файл и ищет в нем строку `%!`, с которой начинаются все PostScript-файлы Если она присутствует, файл передается на принтер без изменений В противном случае вызывается программа `nenscript` для преобразования файла в формат PostScript (предполагается, что файл является текстовым, если попытаться распечатать бинарный файл, например графическое изображение, сценарий завершится неуспешно)

На заметку Сценарий в листинге 9 2 полезен лишь для PostScript-принтеров, хотя его можно приспособить и к принтерам других типов, если вместо программы `nenscript` вызывать Ghostscript

Листинг 9.2. Простейший входной фильтр

```
# /bin/sh
# Проверяем, представлены ли входной файл в формате PostScript
# Если нет, передаем его программе nenscript

read firstline
firstchars= expr  $firstline      \    \)
```

```
if [ "$firstchars" =  %   ]
then
    # Файл PostScript. Передаем на печать строку $firstline
    # и остальную часть файла.
    echo "$firstline" && cat && printf "\004"
    exit 0
else
    # Другой формат. Вызываем программу nenscript.
    (echo "$firstline"; cat) . /usr/local/bin/nenscript  p  \
        && printf "\004"
    Exit 0
fi
```

Строка, начинающаяся с выражения `firstchars=`, содержит два типа одинарных кавычек обратные и прямые. У них разное назначение поэтому не путайте их

Показанный сценарий читает файл из стандартного входного потока и записывает его в стандартный выходной поток. Команда `read firstline` записывает первую строку файла в переменную `$firstline` Далее из этой строки извлекаются два первых символа, которые записываются в переменную `$firstchars`. В строке `if` проверяется, представлен ли файл в формате PostScript. Если первые два символа представляют собой идентификатор PostScript, сценарий записывает в выходной поток первую строку и остальную часть файла в неизменном виде. В противном случае файл передается программе `nenscript`, которая преобразует текстовый файл в формат PostScript

Сохраните сценарий в подходящем каталоге, таком как `/usr/local/bin`, и назовите его, к примеру, `t2ps-filter` Затем задайте права доступа к сценарию

```
# chmod a+x /usr/local/bin/t2ps-filter
```

Далее необходимо модифицировать соответствующую запись файла `/etc/printcap`, чтобы опция `if` ссылалась на созданный сценарий Если ориентироваться на листинг 9 1, то к последней строке нужно добавить еще одну

```
        :sh:\
        :if=/usr/local/bin/t2ps-filter
```

Не забудьте добавить обратную косую черту к бывшей последней строке, иначе добавленная строка будет интерпретирована как начало нового определения, что неправильно.

Перезапустите демон `lpd`, и он подключит к очереди входной фильтр Если теперь попытаться напечатать текстовый файл, "вмешается" программа `nenscript`, которая добавит выделенную полужирным шрифтом строку заголовка с указанием текущего времени и имени файла (в данном примере это будет `stdin`). Естественно, сама программа `nenscript` должна быть инсталлирована. Ее можно найти в разделе `Print` меню `Packages` программы `sysinstall`, как описывается в главе 11

Конфигурирование программы Ghostscript

В некоторых системах фильтр, показанный в листинге 9.2, может быть полезен, но его возможности ограничены. В частности, сценарий предполагает использование PostScript-принтера, хотя это далеко не всегда так Недочет легко исправить, внеся в сценарий указанные выше изменения Более серьезная проблема заключается в том, что в сценарии обрабатываются файлы лишь двух типов PostScript и простой текст. В

противном случае сценарий либо ничего не напечатает либо напечатает "мусор" Решением проблемы является сложный фильтр, содержащий правила идентификации файлов и передающий данные соответствующим программам печати

Если конфигурируется очередь сетевого принтера, который не поддерживает язык PostScript, но обслуживается системой типа FreeBSD с помощью программы Ghostscript, считайте, что это PostScript-принтер. В такой конфигурации программа Ghostscript должна работать на сервере печати, а в клиентских системах она не нужна

Использование программы Ghostscript в роли транслятора

Сложные фильтры используют в качестве транслятора программу Ghostscript. Она входит в состав FreeBSD и является интерпретатором языка PostScript. Будучи вызванной для обработки очереди на печать, программа превращает любой поддерживаемый принтер в PostScript-принтер. Список поддерживаемых принтеров можно найти на Web-узле Linux Printing по адресу `http://www.linuxprinting.org/printer_list.cgi`. Там приведена более полная информация, чем на официальном Web-узле программы (`http://www.cs wisc.edu/~ghost/`), в частности указаны сведения о драйверах, не включенных в пакет Ghostscript

Программа Ghostscript не ограничивается обработкой очередей на печать. Она может принять любой PostScript-файл и создать выходной файл, предназначенный для записи на диск. Базовый синтаксис вызова таков

```
gs [опция[. .]] [файл[...]]
```

Полный список опций приведен на `man` странице команды `gs`. Рассмотрим наиболее важные из них

- **`-sDEVICE=`***имя_устройства* — задает выходное устройство (т.е. драйвер принтера, это не файл устройства FreeBSD). Введите `gs -help`, чтобы получить список доступных устройств. Некоторые названия достаточно неочевидны. Их расшифровка приведена на Web-узле Linux Printing.
- **`r`** *горизонтальное***`x`***вертикальное* — задает разрешающую способность устройства. Например, установка `r 360x720` означает разрешение 360×720 точек на дюйм. Если не указано вертикальное расширение, программа подразумевает, что оно равно горизонтальному. Разрешающая способность должна поддерживаться как самим принтером, так и драйвером Ghostscript.
- **`-sOutputFile=`***имя_файла* — задает выходной файл. Если программа используется для обработки очереди на печать, вместо имени файла будет стоять дефис (`-`), обозначающий стандартный выходной поток

Программа Ghostscript обычно принимает данные из стандартного входного потока. В качестве примера предположим, что имеется принтер Hewlett-Packard LaserJet 4. Показанная ниже команда создает соответствующий выходной файл из PostScript-файла `escher.ps`.

```
$ gs -sDEVICE=ljet4 -r600x600 -sOutputFile=escher.pcl escher ps
```

Такая команда выдает приглашение нажать <Enter> перед обработкой каждой следующей страницы. Для выхода из этого режима введите `quit`. Чтобы отключить вывод приглашений, добавьте перед именем входного файла опции `-dNOPAUSE` и `-dBATCH`. Они важны, если создается пользовательский фильтр, предназначенный для обработки PostScript-файлов

Помимо форматов печати программа Ghostscript поддерживает также распространенные форматы графических файлов а именно: TIFF (tiff24nc и др.), PNG (png16, png256, png16m и др.), JPEG (jpeg), EPS (epswrite), PostScript (pswrite) и PDF (pdfwrite) Может показаться странным — создавать выходной файл PostScript из входного, но иногда это бывает удобно, если, например, PostScript-принтер "зависает" при обработке входного файла из-за нехватки памяти или других проблем

После того как выходной файл создан, направьте его на принтер с помощью ко манды `cat` или `lpr` (последняя может не работать при наличии фильтра). Если ничего не печатается или принтер выдает одну за другой страницы "мусора", попробуйте другой драйвер Если же нет никаких признаков того, что принтер получил задание (индикаторные лампочки не мигают) проверьте соединение с принтером

Настройка сложного фильтра

Для FreeBSD выпущено несколько программ фильтрации Это в сущности те же фильтры, что и сценарий в листинге 9.2, но более сложные. Эти программы умеют распознавать различные типы файлов и обрабатывать их надлежащим образом. Как правило, фильтры вызывают одну из стандартных программ преобразования входных файлов в формат PostScript. После этого вызывается программа Ghostscript, которая транслирует PostScript-файл в формат, понятный принтеру (если используется PostScript-принтер, последний этап пропускается). Когда фильтр сконфигурирован, файл любого поддерживаемого типа печатается с помощью команды `lpr` Наиболее популярны программы фильтрации `apsfilter` и `magicfilter`, обе они доступны в разделе `Print` меню `Packages` программы `sysinstall`.

Порядок работы с программой `magicfilter` необычен. ее конфигурационный файл вызывается так, будто это программа. Такой прием срабатывает потому что конфигурационный файл помечен как исполняемый и начинается с кода, заставляющего FreeBSD запустить программу `magicfilter` в качестве интерпретатора команд Различные конфигурационные файлы программы находятся в каталоге `/usr/local/libexec/magicfilter`. Их имена соответствуют названиям драйверов Ghostscript В частности, фильтр для драйвера `ljet4` называется `ljet4 filter` В качестве примера рассмотрим листинг 9 3. Это уже знакомый вам сценарий, модифицированный для использования программы `magicfilter` при печати на принтере LaserJet 4 Как видите, единственное отличие заключается в вызове конфигурационного файла программы

Листинг 9.3. Пример использования программы `magicfilter` в файле `/etc/printcap`

```
lp hp4000:\
        :lp=/dev/lpt0:\
         br#115200:\
        :sd=/var/spool/lpd/hp4000
        :mx#0:\
        :sh:\
        :if=/usr/local/libexec/magicfilter/ljet4 filter:
```

Если программа `magicfilter` поставляется с фильтром, который подходит для имеющегося принтера, то сконфигурировать принтер будет несложно. Все, что потребуется, — создать очередь на печать. Если же нужный фильтр отсутствует, его придется создать самостоятельно.

На заметку

Программа Ghostscript содержит драйверы для классов принтеров К примеру, многие лазерные принтеры используют тот же формат PCL (Printer Control Language — язык управления печатью), что и принтеры Hewlett-Packard, поэтому драйверы `laserjet`, `ljet2p`, `ljet3`, `ljet4` и т.д (а также соответствующие им конфигурационные файлы программы `magicfilter`) подойдут для принтеров Lexmark, Brother, Okidata и др Дополнительную информацию об этом можно найти на Web-узле Linux Printing (`http://www.linuxprinting.org/printer list.cgi`)

Если возникла необходимость создать пользовательский конфигурационный файл программы `magicfilter`, прежде всего скопируйте существующий фильтр для похожего принтера Не выбирайте файл, имя которого начинается с префикса `ps` или `cps`, поскольку такой фильтр посылает выходные данные в формате PostScript непосредственно на принтер, не вызывая программу Ghostscript Нужные нам фильтры содержат ссылки на программу Ghostscript следующего вида:

```
# PostScript
0 %!     filter /usr/local/bin/gs  q  dSAFER  dNOPAUSE  r600 ↵
-sDEVICE=ljet4 -sOutputFile=
0 \004%' filter /usr/local/bin/gs  q  dSAFER  dNOPAUSE  r600 ↵
-sDEVICE=ljet4  sOutputFile=
```

Измените название драйвера Ghostscript, заданное в опции `sDEVICE`. Многие конфигурационные файлы содержат строки, позволяющие фильтру распознавать файлы, записанные на собственном языке принтера Это дает возможность печатать из приложений наподобие GIMP, которые подготавливают данные в формате конкретного принтера Соответствующие правила нужно модифицировать с учетом имеющегося принтера, что не так-то просто Лучше всего удалить эти строки и использовать стандартные правила обработки PostScript файлов

Иногда приходится создавать альтернативные версии существующих конфигурационных файлов Наиболее распространенный случай — печать с разной разрешающей способностью (меняется значение аргумента `r` в вызове команды `gs`) на одном и том же принтере Для каждого режима нужно организовать отдельную очередь, например `epson360` и `epson720` при печати на струйном принтере Epson с разрешением соответственно 360 и 720 точек на дюйм

По окончании редактирования файла `/etc/printcap` нужно перезапустить демон `lpd`. Попробуйте распечатать с помощью команды `lpr` различные тестовые файлы чтобы определить, какие типы файлов поддерживаются Если программа `magicfilter` не распознает тип файла, этот файл просто удаляется из очереди чтобы не переводить понапрасну бумагу (пользователю `root` будет направлено соответствующее сообщение)

Управление принтером

После того как все указанные выше действия выполнены, очередь на печать готова к использованию. В этом разделе описывается порядок работы с очередью как рядовых пользователей, так и системных администраторов Будет продемонстрировано, как передавать задания на печать, проверять наличие заданий в очереди, удалять их и переупорядочивать, а также менять статус очередей на печать.

Выдача заданий на печать

Ранее уже говорилось о команде `lpr`, которая используется во FreeBSD для постановки заданий в очередь на печать Это очень важная команда заслуживающая от

дельного описания. Нужно также знать, как приложения взаимодействуют с нею ор ганизуя автоматическую печать файлов без вмешательства пользователя

Команда lpr

В упрощенном виде синтаксис команды `lpr` выглядит следующим образом

```
lpr [-Pпринтер] [-#число] [-U пользователь] [-hrms] [файл[   ]]
```

Рассмотрим назначение опций команды

- `-P`*принтер* — задает очередь принтера. В системах с несколькими принтерами это наиболее часто используемая опция. По умолчанию выбирается значение переменной среды `PRINTER` или принтер `lp`, если переменная не определена. Некоторые версии команды требуют, чтобы между опцией `P` и названием очереди не было пробелов но команда включенная в состав FreeBSD терпима к пробелам
- `-#`*число* — заставляет команду вывести указанное число копий документа Напри мер, команда `lpr #4 afile.ps` распечатывает четыре копии файла `afile.ps`
- `-U` *пользователь* FreeBSD следит за тем, кто выдает задания на печать Оп ция `-U`, доступная только пользователям `root` и `daemon` позволяет указать пользователя, закрепленного за очередью
- `h` — подавляет вывод страниц-разделителей.
- `m` — заставляет демон `lpd` послать пользователю уведомление об окончании печати
- `-r` — заставляет команду `lpr` удалить исходный файл по окончании печати. Эта возможность используется сценариями или программами которые генери руют временные файлы для печати.
- `-s`. Обычно демон печати копирует файл задания в каталог спулинга Но если печатается очень большой файл, то это приведет к временному потреблению значительной части дискового пространства которого может не хватить для выполнения задания Опция `-s` заставляет команду не копировать файл а созда вать символическую ссылку на него в целях экономии дискового пространства

Одной комаиядой можно направлять на печать целую группу файлов. Для этого нужно либо перечислить их в командной строке либо воспользоваться подстановоч ными знаками. Если файл не указан, команда `lpr` читает данные из стандартного входного потока Эту возможность удобно использовать в сценариях иян при вызове команды через канал Файлы должны быть представлены в понятном формате. В слу чае простой очереди, для которой нет фильтра, это означает, что формат должен рас познаваться принтером. Если же используется фильтр, он отвечает за распознавание формата файла. В большинстве конфигураций обрабатываются либо текстовые либо PostScript данные

Печать из приложений

Некоторые программы позволяют направлять данные на печать прямо из них, ми нуя вызов команды `lpr`. Такие программы генерируют PostScript-данные и сами пе редают их команде `lpr`, по каналу либо через временные файлы в каталоге `/tmp`. Ме ханизм взаимодействия с пользователем зависит от программы Некоторые програм мы отображают диалоговое окно, в котором можно ввести команду печати Обычно часть команды уже подготовлена, ее нужно лишь модифицировать или дополнить. К примеру, на рис. 9.2 изображено диалоговое окно графической программы `xv` В поле ввода задана команда `lpr` без каких-либо опций Можно направить задание на кон-

кретный принтер с помощью опции -Pпринтер или воспользоваться вообще другой командой, например пользовательским сценарием печати или утилитой smbprint пакета Samba

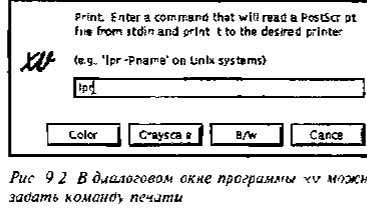


Рис. 9.2. В диалоговом окне программы xv можно задать команду печати

Интерфейсы других программ более понятны пользователям. Чтобы идентифицировать принтер, такие программы проверяют файл /etc/printcap или свои собственные конфигурационные файлы. На рис. 9.3 изображено окно настроек печати в программе OpenOffice.org. В этом окне можно выбрать принтер (список Name), задать число копий и т.д. Диалоговые окна подобного рода часто содержат настройки, специфичные для конкретного приложения. Например, переключатели группы Print range на рис. 9.3 позволяют указать, какие страницы документа должны быть распечатаны.

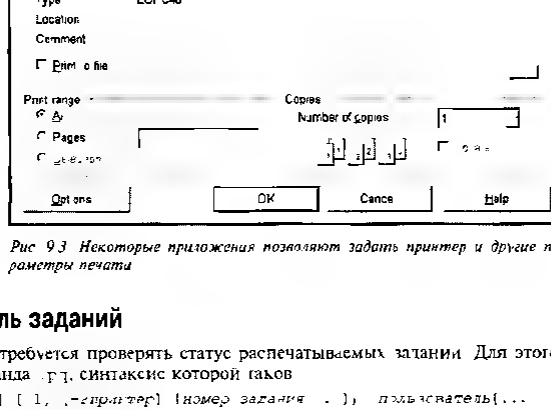


Рис. 9.3. Некоторые приложения позволяют задать принтер и другие параметры печати

Контроль заданий

Часто требуется проверять статус распечатываемых заданий. Для этого предназначена команда lpq, синтаксис которой таков:

```
lpq [-a] [-l] [-Pпринтер] [номер задания ...] [пользователь|...]
```

Опция -Pпринтер имеет такое же назначение, как и в команде lpr: она задает очередь на печать. Остальные опции интерпретируются следующим образом:

- **-a** Если нужно просмотреть содержимое всех локальных очередей укажите опцию -a вместо -P*принтер*.
- -l — заставляет команду выводить дополнительную информацию о заданиях.
- *номер задания* Обычно команда сообщает обо всех заданиях, находящихся в очереди. Если указать номера заданий, команда проверит только их
- *пользователь*. Можно получить информацию о заданиях конкретного пользователя.

Если очередь пуста команда `lpq` выдаст сообщение `no entries` В противном случае результат будет примерно таким:

```
$ lpq
waiting for lp to become ready  offline?)
Rank   Owner      Job  Files                                 Total Size
1st    rodsmith   35   fstab-copy                            450 bytes
2nd    rodsmith   37    ..hostscript/6 52/examples/escher ps 10520 bytes
3rd    rodsmith   38   standard input                        1364390 bytes
```

В системе LPRng или CUPS формат вывода будет другим

Как видите команда указывает ранг задания (порядок в соответствии с которым оно будет напечатано), имя владельца, идентификатор задания, исходный файл и размер задания. Самая первая строка (`waiting for lp to become ready`) говорит о том, что принтер не отвечает Это сообщение может появиться, если принтер выключен, в нем застряла или отсутствует бумага либо возникла другая проблема, препятствующая печати. В таком случае демон `lpd` перестает посылать задания на принтер и переходит в режим ожидания. Идентификатор задания особенно важен, поскольку используется в других командах управления печатью.

В колонке `Total Size` приведен размер задания в байтах Нужно понимать, что это значение не связано с размером распечатки В данном случае задание номер 35 (`fstab copy`) — это небольшой текстовый файл, задание номер 37 (`escher.ps`) это PostScript-файл генерирующий одну страницу выходных данных, а задание номер 38 (берется из стандартного входного потока) представляет собой одностраничную графическую распечатку Несмотря на разницу в размерах во всех трех случаях будет напечатана одна страница

Удаление заданий

Если нужно отменить задание, направленное на печать удалите его из очереди с помощью команды `lprm`. Ее синтаксис таков.

```
lprm [-Pпринтер] [ ] [номер_задания[.. ]] [пользователь[. .]]
```

Опция P*принтер* задает очередь. Чтобы узнать номер задания, выполнните команду `lpq` Таким образом рядовой пользователь может ввести следующую команду:

```
$ lprm 35 38
```

Эта команда удаляет задания номер 35 и 38 из стандартной очереди. Пользователю разрешается удалять лишь те задания, которые принадлежат ему, и только суперпользователь может указать имя другого пользователя Опция ' ' означает удаление всех заданий текущего пользователя

```
$ lprm
```

Для суперпользователя эта команда трактуется иначе она удаляет все задания из очереди независимо от их принадлежности

Переупорядочивание заданий

Иногда приходится менять порядок заданий в очереди. Например в очереди мо жет находиться несколько довольно крупных заданий, но только что поступившее за дание имеет более высокую важность и должно быть напечатано раньше В этом слу чае на помощь приходит утилита lpc Синтаксис ее вызова прост

```
lpc [команда [аргумент[...],]
```

Если не указать команду, утилита перейдет в интерактивный режим в котором по приглашению lpc> можно вводить нужные команды Всего поддерживается более де сятка команд (введите **lpc help**, чтобы получить краткую сводку доступных команд) Команда, изменяющая порядок заданий в очереди, называется topq. В качестве аргу ментов она принимает название очереди и один или несколько идентификаторов за даний либо имя пользователя. Эта команда перемещает определенные задания либо все задания указанного пользователя в начало очереди Вот как можно переместить задание номер 38 в начало очереди hp4000

```
# lpc topq hp4000 38
```

У этой команды есть два важных ограничения

- Утилита lpc доступна только суперпользователю Рядовые пользователи не могут изменять порядок заданий, даже собственных.
- Перемещаемое задание начнет печататься *после* текущего Если очередь связана с принтером, у которого большой аппаратный буфер, или с сетевым принте ром, то изменение порядка заданий может не дать результата из-за того, что задания уже покинули локальную системную очередь

Изменение статуса очереди

Утилита lpc является рабочим инструментом администратора Она может не толь ко изменять порядок заданий в очереди, но и полностью отключать (а также повторно включать) очередь. Есть несколько взаимосвязанных пар команд

- **abort/start**. Команда abort отключает демон lpd от заданной очереди и не позволяет команде lpr запустить новый экземпляр демона. Таким образом она прекращает печать заданий, находящихся в очереди, хотя сами они не уда ляются. Возобновить печать позволяет команда start В некоторых ситуациях удобна также команда restart, которая останавливает, а затем возобновляет печать Это может понадобиться, если обработчик очереди по какой-то причи не "завис"
- **down/up**. Команда down останавливает печать и позволяет задать сообщение отображаемое командой lpq. Например, если ввести **lpc down hp4000 Printer is out of toner**, печать будет отменена, а после ввода команды **lpq -Php4000** пользователь увидит сообщение Printer is out of toner (закончился тонер). Задания остаются в очереди, но не печатаются до тех пор, пока не будет введена команда -p.
- **disable/enable** Команда disable запрещает команде lpr принимать новые задания, адресованные указанной очереди, при этом печать заданий, уже нахо дящихся в очереди, не отменяется и не откладывается. Чтобы возобновить прием заданий на печать введите команду enable

Совет Если нужно временно отключить очередь, с которой работают пользователи, введите команду `disable`, подождите, пока будут завершены текущие задания, а затем с помощью команды `down` сообщите пользователям, почему очередь стала недоступной. Конечно, в случае серьезной поломки принтера нужно сразу воспользоваться командой `abort` или `down`.

Во всех этих случаях в строке вызова утилиты `lpc` сначала указывается необходимая команда, а затем название очереди или ключевое слово `all`, если команда касается всех очередей. Например, следующая команда прекращает прием заданий для всех очередей, не останавливая обработку уже имеющихся заданий

```
# lpc disable all
```

Контроль доступа к сетевому принтеру

Как уже указывалось ранее, система печати BSD LPD поддерживает сетевую печать. В разделе "Создание очереди на печать" было показано, как сконфигурировать локальную очередь для переадресации запросов на BSD-сервер печати. Осталось выяснить, как настроить сам сервер, чтобы он мог принимать задания на печать от других систем. Если на сервере установлена FreeBSD или другая система, понимающая протоколы BSD LPD, то конфигурирование не представляет трудностей: достаточно включить режим локальной печати на сервере, а затем отредактировать один файл.

На заметку В этом разделе описывается конфигурация компьютера, принимающего задания с использованием протоколов BSD LPD. Системы печати LPRng и CUPS поддерживают эти протоколы на стороне клиента. Для серверов LPRng и CUPS приводимые ниже инструкции неприменимы. Кроме того, если клиентская система не относится к семейству UNIX, то лучше реализовать ее собственные протоколы печати. Для Windows и OS/2 это делает пакет Samba (описан в главе 18); пакет Netatalk (`http://netatalk.sourceforge.net`) включает в себя сервер печати для клиентов Mac OS.

По умолчанию демон `lpd` принимает запросы только из локальной системы. Чтобы снять это ограничение, нужно в файле `/etc/hosts.lpd` указать список компьютеров, от которых могут поступать задания. Разрешается указывать имя компьютера (локальное или доменное, если не задан доменный компонент имени, демон `lpd` подразумевает, что клиент находится в домене сервера) или его IP-адрес. В листинге 9.4 показан файл `hosts.lpd`, в котором доступ к серверу печати разрешается компьютеру `chaucer`, находящемуся в домене сервера, а также компьютерам `shakespeare.threeroomco.com` и `172.19.45.2001`.

Листинг 9.4. Образец файла `/etc/hosts.lpd`

```
chaucer
shakespeare.threeroomco.com
172.19.45.2001
```

Для того чтобы разрешить другим компьютерам доступ к системе, можно также использовать файл `/etc/hosts.equiv`, однако он применяется не только программами печати, но и различными сетевыми утилитами. В целях безопасности рекомендуется конфигурировать серверы отдельно друг от друга. В главах 17 и 29 будет подробнее рассказано о том, как предотвратить несанкционированный доступ в систему.

Резюме

Как и другие современные операционные системы, FreeBSD управляет печатью посредством очередей. Такая схема позволяет системе планировать обработку заданий, что имеет большое значение в многозадачных системах и критически важно для многопользовательских систем и серверов печати. А вот драйверы принтеров реализуются во FreeBSD не так, как в других системах. Во FreeBSD используются программы, генерирующие текстовые или PostScript-данные для печати, а затем применяются фильтры, которые преобразуют эти данные в формат, понятный используемому принтеру. Очереди на печать конфигурируются посредством файла `/etc/printcap`, в котором указывается название очереди, возможности принтера, с которым она связана и применяемый фильтр. Если печать выполняется не на PostScript-принтерах, фильтры вызывают программу Ghostscript, которая преобразует PostScript-данные в соответствующий формат. Существует целый ряд команд, предназначенных для управления очередями, включая `lpr`, `lpq`, `lprm` и `lpc`. Некоторые приложения реализуют графические интерфейсы доступа к этим командам, особенно к `lpr`, которая ставит задания в очередь. Система печати BSD LPD изначально включала средства поддержки сетевой печати, поэтому, чтобы сконфигурировать FreeBSD в качестве сервера печати достаточно включить список разрешенных клиентов в файл `/etc/hosts.lpd`.

Полный справочник по

Глава 10

Управление учетными записями

Учетные записи играют главную роль в организации работы пользователей и защиты системы, поэтому управление ими — одна из важнейших задач администратора FreeBSD. Как правило, каждый пользователь имеет одну учетную запись, хотя бывают и исключения. Некоторые системные и административные учетные записи либо не используются никем, кроме самой системы, либо доступны только системным администраторам. В предыдущих главах уже рассказывалось о важных учетных записях (например `root`), здесь же эта тема раскрывается более детально. Будет показано, как создавать, модифицировать и удалять учетные записи пользователей и групп. В завершение главы мы поговорим о взаимосвязи учетных записей с безопасностью системы.

На заметку Управление учетными записями тесно связано с заданием владельцев файлов и прав доступа к файлам. Об этом речь шла в главе 8.

Роль учетных записей во FreeBSD

Прежде чем приступать к администрированию учетных записей, нужно понять для чего они существуют, как используются и как реализуются. Структура учетных записей должна соответствовать потребностям системы, а у FreeBSD есть свои особенности.

Упорядочение работы пользователей

Учетные записи нужны прежде всего для удобства пользователей. В стандартной инсталляции ряда операционных систем по умолчанию загружается единая настольная среда. Пользователь может модифицировать параметры этой среды и записывать данные в произвольный каталог. К таким операционным системам относятся DOS, Windows 9x/Me, OS/2, Mac OS Classic и BeOS (в некоторых из них есть конфигурационные параметры, позволяющие создавать учетные записи). Такой тип конфигурации подходит для однопользовательской системы, поскольку он сокращает время и усилия, необходимые для освоения компьютера. Пользователю достаточно включить компьютер и подождать, пока загрузится настольная среда.

В многопользовательской среде описанный подход неудобен. Предположим, два пользователя — Пол и Мэри — работают за одним компьютером. Если Пол предпочитает однотонный фон экрана, а Мэри нравится использовать фоновые фотографии, то кому-то из них придется уступить. Решить проблему можно с помощью учетных записей, которые позволяют хранить установки отдельно для каждого пользователя. Система будет выбирать установки пользователя, регистрирующегося в данный момент, и все останутся довольны.

На заметку Пользовательские установки поддерживаются во многих программах FreeBSD, а не только в графических оболочках. К таким программам относятся почтовые клиенты, Web-броузеры, текстовые процессоры и даже интерпретаторы команд текстового режима. Об этом рассказывалось, в частности, в главе 6.

Другим преимуществом учетных записей является то, что для каждой из них во FreeBSD выделяется своя область хранения файлов, называемая *начальным каталогом*. Обычно это подкаталог каталога `/home`, называемый по имени пользователя, например `/home/marie` или `/home/paul`. Благодаря наличию отдельного каталога пользова

тели получают возможность хранить свои файлы так, как считают нужным В зависимости от установок, пользователи могут иметь или не иметь доступ к чужим начальным каталогам

Безопасность системы

Размещение пользовательских файлов в отдельных каталогах выгодно не только с точки зрения удобства, но и с точки зрения безопасности Благодаря механизму прав доступа пользователи могут контролировать, кто имеет право обращаться к начальным каталогам, подкаталогам и отдельным файлам и как именно. Например, если Пол работает над проектом, к которому Мэри не имеет отношения, он может задать такие права к соответствующим каталогам и файлам чтобы для Мэри они были недоступны Это чрезвычайно важно в многопользовательской операционной системе

Помимо контроля доступа на уровне отдельных пользователей учетные записи обеспечивают безопасность системы в целом Большинство файлов самой FreeBSD доступно для чтения, а многие из них — и для выполнения, любому пользователю. В то же время почти все системные файлы и каталоги защищены от записи, чтобы никто из пользователей, кроме системного ахминистратора (учетная запись `root`), не мог удалить их. Эта особенность позволяет относительно неподготовленным пользователям работать с FreeBSD в относительно открытой среде Применение Windows 9x/Me или Mac OS Classic в открытой среде часто вызывает проблемы, поскольку пользователи по неопытности или злонамеренно могут повредить системную конфигурацию. Во FreeBSD таких проблем не возникает

Когда во FreeBSD работает один пользователь у него может возникнуть соблазн отказаться от обычной учетной записи и рагистрироваться под именем `root`. Никогда так не делайте! На первый взгляд это заманчиво — пользоваться преимуществами привилегированного режима, не прибегая к помощи команды `su` Но такой подход очень опасен. Во FreeBSD практически нет средств защиты от действий суперпользователя. Обычная команда, например `rm`, введенная с ошибкой, может привести к катастрофическим последствиям. В режиме суперпользователя нужно выполнять лишь самые необходимые действия

Группы пользователей

Учетные записи — мощное средство разграничения доступа, но во многих ситуациях их возможностей недостаточно Часто требуется обеспечивать безопасность на уровне групп пользователей. *Группа* — это именованное объединение нескольких учетных записей Например, пользователи `paul` и `marie` могут входить группу `physics`, а `jane` и `charles` — в группу `biology`. Допускается, чтобы один пользователь был членом нескольких групп.

Группы используются в основном для контроля доступа к файлам как описывалось в главе 8 В разделе "Безопасность учетных записей" будет рассказано о возможных конфигурациях групп

Одна группа особенно важна: `wheel` Ее члены (их число должно быть строго ограничено) имеют право с помощью команды `su` получать привилегии других пользователей, включая `root`. Только им должен быть известен пароль суперпользователя Член группы `wheel` может войти в систему под своим обычным именем, ввести `su` и пароль суперпользователя и приступить к администрированию системы Всем остальным пользователям запрещен доступ к команде `su`. Это снижает вероятность взлома системы, даже если злоумышленник каким-то образом узнал пароль суперпользователя Таким образом, группа `wheel` является важным элементом защиты системы

Совет Чтобы выполнять административные действия член группы wheel должен зарегистрироваться в системе под своим обычным именем а затем перейти в режим суперпользователя с помощью команды su. Это позволит зафиксировать введенные команды в системных журнальных файлах, чего не произойдет, если войти в систему непосредственно под именем root. Таким образом, журнальные файлы являются средством контроля за действиями администраторов.

Имена и идентификаторы пользователей и групп

Имена пользователей и групп существуют лишь для удобства самих пользователей. FreeBSD преобразует эти имена в идентификаторы UID (User ID) и GID (Group ID), которые хранятся вместе с другими метаданными файла (данными, описывающими файл, в отличие от его содержимого), такими как время его создания и размер. Идентификаторы UID и GID используются также для обозначения владельцев процессов.

Во FreeBSD существует механизм преобразования идентификаторов в понятные имена. Вот почему при выводе расширенного листинга команда ls отображает имена пользователей и групп а не их идентификаторы:

```
$ ls -l license.txt
-rw-------  1 paul  users      356 Mar  2 2002 license.txt
```

На заметку Если указать опцию n команда ls отобразит идентификаторы а не имена

Обычно существует однозначное соответствие между идентификаторами и именами Но допускается создавать учетные записи, в которых одному идентификатору сопоставлено несколько имен. Такие конфигурации, впрочем, довольно непривычны и как правило нежелательны

Внимание Хакеры иногда прибегают к следующему трюку: создается учетная запись с обычным, не вызывающим подозрения именем пользователя но идентификатором 0. Это специальный идентификатор соответствующий учетной записи root. Таким образом, "новая" учетная запись хакера в действительности позволяет ему стать суперпользователем, хоть и под другим именем Помните о возможных злоупотреблениях. Если в файле паролей обнаруживается учетная запись с идентификатором 0, имя которой не root, система почти наверняка была взломана

По умолчанию идентификаторы пользователей и групп FreeBSD назначаются начиная с 1000. В других системах нумерация ведется от 100 Поскольку эти идентификаторы связаны с файлами, при копировании файлов из одной системы в другую нужно убедиться в том, что пользователь имеет тот же идентификатор в обеих системах иначе права доступа будут нарушены. Значение GID тоже должно соответствовать эквивалентной группе. Имена учетных записей роли не играют, если за ними стоит один пользователь. Эти соображения актуальны, когда речь идет о копировании посредством дискет или Zip-дисков формата FFS (Fast File System), а также о пересылке файлов средствами NFS (Network File System). Если же файловая система не поддерживает идентификаторы пользователей и групп, например FAT то права доступа в любом случае будут изменены То же самое произойдет и в случае протокола FTP

Структуры данных, связанные с учетными записями

Учетные записи состоят из набора взаимосвязанных структур данных хранящихся в различных файлах и каталогах. Перечислим важнейшие из них

- **/etc/passwd** — это текстовый файл, содержащий основную информацию обо всех пользовательских учетных записях. Для каждой записи указаны идентификатор и имя пользователя, идентификатор ствндартной группы, начальный каталог и интерпретатор команд. Ранее здесь же хранились пользовательские пароли, но современные системы более защищены. Этот файл доступен для чтения всем пользователям компьютера
- **/etc/master.passwd** — здесь хранится в основном та же информация, что и в файле /etc/passwd, плюс хеш-коды (односторонние шифры) пользовательских паролей и ряд других сведений. Из-за наличия паролей, пусть и в зашифрованном виде, этот файл доступен для чтения только пользователю root.
- **Бинарные файлы паролей.** Текстовые файлы /etc/passwd и /etc/master.passwd подходят для небольших систем, но когда в файле содержатся сотни или тысячи записей, текстовый поиск в нем занимает слишком много времени. С учетом этого во FreeBSD предусмотрены специальные бинарные файлы паролей: /etc/pwd.db и /etc/spwd.db. Благодаря им ускоряется поиск информации о пользователях
- **Начальный каталог** - является важным компонентом учетной записи пользователя, поскольку хранит пользовательские файлы данных. Местонахождение этого каталога определяется в файлах /etc/passwd и /etc/master.passwd.
- **Локальные конфигурационные файлы.** В пользовательских каталогах часто содержатся файлы, определяющие стандартные установки пользовательской среды. К примеру, конфигурационные файлы интерпретатора команд находятся в начальном каталоге пользователя (см. главу 6).
- **Серверные конфигурационные файлы.** Некоторые серверы тесно связаны с учетными записями используемыми для регулирования доступа к этим серверам. Было бы преувеличением считать такие серверы компонентами учетных записей, просто о них нужно помнить при добавлении, удалении и модификации учетных записей

В последующих разделах будет рассказано о том, как редактировать эти файлы, изменяя характеристики учетных записей. Схожие конфигурационные файлы существуют и для групп, но они несколько проще. Наиболее важный из них — /etc/group, содержащий определения групп. Каждое такое определение включает в себя название группы, ее идентификатор и список имен пользователей. У группы нет начального каталога, но можно создавать каталоги, открытые для доступа всем членам группы.

Создание учетных записей

В процессе инсталляции FreeBSD разрешается создать начальные учетные записи (см. главу 2). Система создает также ряд специальных учетных записей (их идентификаторы пользователей меньше 100, за исключением одной — nobody: ее идентификатор равен 65534). Однако в ходе эксплуатации системы, особенно если это файловый или почтовый сервер либо сервер дистанционного доступа, может возникнуть необходимость в дополнительных учетных записях

Для добавления учетных записей пользователей обычно применяют программу `sysinstall` или утилиту командной строки `adduser`. Можно и напрямую редактировать файлы паролей, но это наименее удобный способ.

Правила именования пользователей

Прежде чем создавать учетные записи, следует выяснить, какого вида имена можно назначать им. Во FreeBSD поддерживаются имена пользователей, состоящие из строчных букв, цифр, знаков подчеркивания (_) и дефисов (-). Максимальная длина имени — 16 символов, но в некоторых программах для отображения имен предусмотрено 8 знакомест, поэтому длинные имена могут усекаться или же нарушать выравнивание последующих столбцов.

На заметку

В некоторых UNIX-системах допускаются более длинные имена, а также имена с буквами смешанного регистра и расширенными символами (например, знаками пунктуации). Во FreeBSD подобные имена можно создавать путем прямого редактирования файла `/etc/master.passwd` с последующим выполнением команды `pwd mkdb`, но это может вызвать проблемы при работе с утилитами, которые придерживаются стандартных правил именования пользователей.

Имена пользователей FreeBSD чувствительны к регистру символов. Если, к примеру, пользователь `marie` попытается войти в систему под именем `Marie`, система откажется принять пароль. Утилиты управления учетными записями тоже требуют соблюдения регистра символов. Исключением из правила является электронная почта: если в качестве получателя письма указать `Marie`, оно будет доставлено пользователю `marie`.

В остальном имена пользователей выбираются произвольным образом. Чаще всего они соответствуют настоящим именам или фамилиям. Например, у пользователя Мэри Джонс может быть регистрационное имя `marie`, `jones`, `mjones` или что-то в этом роде. Если у пользователя слишком длинные имя и фамилия, то при выборе регистрационного имени их приходится усекать. Некоторые администраторы придерживаются строгих схем соответствия между реальными и регистрационными именами, но, как правило, пользователи выбирают имена по своему усмотрению. В любом случае у каждой организации свои правила выбора имен.

Использование программы sysinstall

Программа `sysinstall` позволяет добавлять новых пользователей в систему. В общих чертах этот процесс описывался в главе 2. Ниже мы шаг за шагом рассмотрим, как создается учетная запись.

1. Перейдите в режим суперпользователя и введите **`/stand/sysinstall`**. Эта команда запускает программу `sysinstall`.
2. Выберите в основном меню программы пункт `Configure`. В результате будет вызвано меню конфигурирования FreeBSD.
3. Выберите пункт `User Manager`, чтобы перейти в меню `User and Group Management`.
4. Выберите пункт `User`, чтобы отобразить форму `User and Group Management` (рис. 10.1). Некоторые поля формы, в частности `UID` и `Login Shell`, уже будут заполнены.

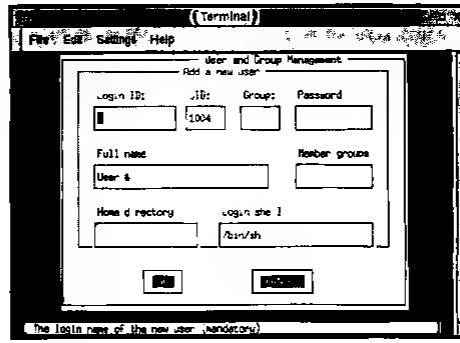


Рис 10 1 Форма User and Group Management предназначена для создания учетных записей

5. Введите имя пользователя в поле Login ID Нажмите <Tab> или <Enter>, после чего курсор переместится в поле UID, а в поле Home Directory будет записано сгенерированное имя начального каталога, соответствующее введенному имени пользователя, например /home/marie.
6. Если необходимо, поменяйте идентификатор пользователя в поле UID Программа sysinstall выбирает первый доступный идентификатор
7. Введите название стандартной группы в поле Group. Эта группа должна существовать, в противном случае программа sysinstall не сможет создать учетную запись. Если пользователь должен иметь доступ к команде su задайте группу wheel. Процесс создания групп описан на с 251
8. Введите пароль в поле Password. Грамотный выбор пароля — один из решающих факторов для безопасности системы О выборе безопасных паролей будет подробно рассказываться в главе 29 Паролем не должно быть слово из словаря или имя (особенно имя, ассоциированное с самим пользователем, например имя жены). Пароль должен содержать прописные и строчные буквы, желательно также цифры и знаки препинания.
9. В поле Full Name задается произвольная информация о пользователе, отображаемая некоторыми утилитами Обычно здесь указывают имя и фамилию. Текст разделяемый запятыми, по-особому интерпретируется утилитой finger. Таким способом можно задать имя и фамилию, служебный адрес служебный и домашний телефоны
10. Если пользователь должен входить в несколько групп, а не только в ту, что задана в п 7, укажите их в поле Member groups. Список групп разделяется запятыми Это поле может быть пустым
11. В поле Home directory можно поменять имя начального каталога выбираемое по умолчанию.
12. В поле Login shell задается интерпретатор команд пользователя Он должен присутствовать в списке разрешенных интерпретаторов (хранится в файле /etc/shells), иначе система выдаст предупреждение

13 Переместите курсор на поле OK и нажмите <Enter> Система создаст учетную запись на основе предоставленной информации, после чего снова отобразится меню User and Group Management Если требуется создать дополнительные учетные записи, повторите пп. 4—13 В противном случае дважды выберите Cancel а затем — Exit Install, чтобы выйти из программы sysinstall

Использование утилиты adduser

Описанная выше процедура подходит для большинства случаев Программа sysinstall выдает подсказки по поводу содержимого каждого поля что существенно упрощает ее применение. Но если требуется создать много учетных записей, то удобнее написать сценарий, автоматизирующий эту процедуру. В такой ситуации на помощь приходит утилита adduser. Она выдает запросы на ввод необходимой информации, работая со стандартными потоками ввода-вывода, что позволяет вызывать ее из сценариев.

При первом вызове утилита adduser пытается обратиться к конфигурационному файлу /etc/adduser.conf, который не существует в стандартной инсталляции. В результате утилита выдает все предусмотренные запросы чтобы определить установки по умолчанию В большинстве случаев достаточно нажать <Enter>, чтобы принять предлагаемый вариант Впоследствии файл adduser.conf можно отредактировать или удалить и запустить утилиту adduser повторно.

В первую очередь утилита запрашивает имя пользователя и общую информацию о нем

```
# adduser
Use option ``-verbose'' if you want to see more warnings and
questions or try to repair bugs.

Enter username [a-z0-9_-]. marie
Enter full name []: Marie Curie
```

Как указывалось ранее, регистрационное имя может состоять из букв нижнего регистра и цифр. В ответ на запрос Enter full name обычно вводят имя и фамилию но по сути это может быть произвольная информация. Далее программа задает вопросы касающиеся выбора интерпретатора команд, начального каталога идентификатора пользователя и групп:

```
Enter shell bash csh date ksh no pash sh tcsh zsh [sh]  ksh
Enter home directory (full path) [/home/marie]
Uid [1003]:
Enter login class: default []:
Login group marie [marie]:
Login group is ``marie''. Invite marie into other groups  gueet no
[no]: users
```

Если предложенный вариант вас устраивает, просто нажмите <Enter>. В рассматриваемом примере сделаны две нестандартные установки Во-первых, выбран интерпретатор ksh, а во-вторых, пользователь включен в дополнительную группу users. Здесь следует сделать пояснение По умолчанию для каждого пользователя утилита adduser создает специальную группу, название которой совпадает с его регистрационным именем Собственно говоря, этот пользователь чаще всего является единственным членом такой группы. Это удобно с точки зрения контроля доступа к начальному каталогу пользователя Другие группы используются в основном для организации совместного доступа к данным

На следующем этапе задаются вопросы касающиеся выбора пароля

```
Use password-based authentication (y/n  [y]:
Use an empty password (y/n) [n]
Enter password [ :
Enter password again []:
Enable account password at creation  y/n   y]
```

В данном случае для всех установок, кроме собственно паролей, выбраны значения по умолчанию. Пароли здесь не показаны, так как они не отображаются на экране. Один и тот же пароль нужно ввести дважды, чтобы случайная опечатка не привела к блокированию учетной записи. После этого утилита попросит подтвердить введенную информацию:

```
Name:     marie
Password: ***
Fullname: Marie Curie
Uid:      1003
Gid:      1003 (marie)
Class:
Groups:   marie users
HOME:     /home/marie
Shell:    /usr/local/bin/ksh
OK? (y/n) [y]: y
```

Если ввести n, утилита повторит всю процедуру с самого начала. В противном случае будет создана учетная запись. В конце утилита предложит создать почтовое сообщение, посылаемое новому пользователю:

```
Added user ``marie''
Send messages to ``marie'' and no root second mail address
[no]: charlesd@pangaea.edu

Marie Curie

your account 'marie' was created
Have fun!

See also chpass(1), finger(1) passwd(1

Add anything to default message (y/n) [n] n
Send message (y/n) [y]: y
Add another user? (y/n) [y]: n
```

Система спрашивает, нужно ли отправить копию сообщения по другим адресам. Стандартный ответ — no. Если ввести root, письмо получит суперпользователь. В данном примере письмо посылается по адресу charlesd@pangaea.edu. Далее утилита отображает само сообщение, которое не слишком информативно. В конце концов, любой пользователь, вошедший в систему, и так знает, что у него есть учетная запись. Сообщение хранится в файле /etc/adduser.messages, который разрешается редактировать. Такое сообщение увидят все новые пользователи системы. Сообщение можно изменять и для каждого пользователя в отдельности. При желании можно вообще не отправлять сообщение. В последней строке утилита спрашивает, нужно ли создавать новую учетную запись. Если ответить no, утилита завершит работу.

Утилита adduser принимает различные опции, перекрывающие установки файла /etc/adduser.conf. Например, опция -home *каталог* задает первый компонент пути к начальному каталогу (обычно /home). Необходимость в таких опциях возникает редко. Исключение составляет опция -config_create, которая заставляет утилиту создать новый конфигурационный файл.

Модификация учетных записей

В идеале о созданной учетной записи можно забыть. Но так бывает далеко не всегда. Чаще всего приходится менять пароль. Это должно происходить достаточно регулярно, чтобы потенциальный ущерб от взлома пароля был сведен к минимуму. Периодичность зависит от конкретных условий. В высоконадежных системах могут

использоваться одноразовые пароли, устаревающие сразу после применения В ос тальных случаях достаточно менять пароль раз в несколько месяцев. Важно понимать как функционируют пароли и как модифицировать не только их но и другие пара метры учетных записей

Скрытые пароли

Традиционно пароли UNIX хранились в файле `/etc/passwd` наряду с прочей ин формацией Доступ к этому файлу нужен многим программам Например, программа `finger`, которая отображает информацию о пользователях системы, должна иметь возможность прочитать из него настоящее имя пользователя По этой причине файл доступен для чтения всем пользователям Такой подход считался приемлемым, по скольку пароли хешировались К сожалению, с появлением высокопроизводительных компьютеров и жестких дисков большой емкости стало возможным хранить хеширо ванные версии целых словарей и сравнивать их с содержимым файла `/etc/passwd` Это позволяло быстро выявлять плохо подобранные пароли.

Частичным решением проблемы является принудительный выбор более надежных паролей, но этого трудно добиться Альтернативный подход заключается в перемеще нии паролей из файла `/etc/passwd` в другой, более защищенный файл Во FreeBSD он называется `/etc/master.passwd`, а его бинарный вариант — `/etc/spwd.db` В них хранится та же информация, что и в аналогичных незащищенных файлах, но она доступна только пользователю `root` В самом файле `/etc/passwd` в поле пароля сто ит звездочка, указывающая на то, что пароль хранится отдельно

Такие пароли называются *скрытыми*. В разных системах они реализованы по разному, но суть всегда в том, что информация о паролях хранится в отдельных фай лах, доступных лишь суперпользователю и его программам. Остальным программам по-прежнему предоставляется доступ к файлу `/etc/passwd`. Этот подход тоже не обеспечивает стопроцентную безопасность. Если хакер получает хотя бы частичный доступ к учетной записи `root`, он может украсть файл скрытых паролей и попытаться взломать его Тем не менее скрытые пароли являются важным средством защиты FreeBSD

Команда passwd

Первоначально пароль вводится при создании учетной записи Время от времени его приходится менять, чтобы снизить риск взлома. Для этого предназначена команда `passwd` Ее синтаксис очень прост:

```
passwd [-l] [пользователь]
```

Если система сконфигурирована как часть инфраструктуры Kerberos, командв ме няет пароль Kerberos Опция `-l` заставляет ее менять только локальный пароль.

На заметку

Kerberos — это распределенная система аутентификации, популярная в сетях с большим числом серверов и в сетях, где пользователи могут регистрироваться на множестве рабочих станций Дополнительную информацию о системе можно найти по адресу `http://web.mit.edu/kerberos/www/`

Только суперпользователь имест право задавать имя пользователя, чеи пароль дол жен быть изменен Поэтому в большинстве случаев схема применения команды вы глядит так

```
$ passwd
Changing local password for marie
Old password:
```

```
New password:
Retype new password:
passwd: updating the database
passwd: done
```

В целях безопасности команда запрашивает старый пароль, чтобы злоумышленник не мог, к примеру, воспользоваться терминалом, оставленным без присмотра Затем команда дважды запрашивает новый пароль и сообщает о том, удалось ли выполнить обновление Ни один из паролей не отображается на экране.

Если команда `passwd` вызывается суперпользователем, старый пароль не запрашивается Это позволяет суперпользователю менять любой пароль, что особенно удобно в том случае, когда кто нибудь из пользователей забыл пароль и просит администратора помочь ему

Команда chpass

Другая команда управления учетными записями называется `chpass` Она позволяет изменять параметры учетной записи, включая пароль, срок действия записи и стандартный интерпретатор команд. Синтаксис команды таков:

```
chpass [ a список] [-p зашифрованный_пароль] [ e срок_действия] ↵
       [ s интерпретатор] [пользователь]
```

На заметку

Команда `chpass` поддерживает также опции, связанные с управлением учетными записями NIS (Network Information System). NIS — это распределенная административная СУБД, напоминающая Kerberos Дополнительная информация об этих опциях приведена на man-странице команды

Большинство опций команды доступно лишь суперпользователю Ниже описано их назначение

- **-a** *список* — позволяет суперпользователю задавать полную запись в формате файла `/etc/passwd`
- **-p** *зашифрованный_пароль* — доступна только суперпользователю и позволяет ему менять пароль, который предварительно должен быть зашифрован с помощью программы `crypt` Обычно опция применяется в сценариях, которые вызывают программу `crypt` и передают полученный результат команде `chpass`
- **-e** *срок действия* — доступна только суперпользователю и позволяет ему задавать срок действия учетной записи. Значение даты вводится в кавычках в формате "*месяц день год*", например "March 20 2003". После указанной даты доступ к учетной записи запрещен.
- **-s** *интерпретатор* — позволяет сменить стандартный интерпретатор команд
- *пользователь* — доступна только суперпользователю и задает модифицируемую учетную запись

Чаще всего команда `chpass` вызывается без аргументов или с аргументом *пользователь* В этом случае запускается редактор, в котором можно изменить параметры учетной записи, в том числе те, для которых нет аргументов командной строки На рис. 10 2 изображено окно сеанса редактирования (используется редактор `vi`). Если команда вводится рядовым пользователем, изменять разрешается лишь часть полей

Многие поля уже были описаны Ниже указаны лишь основные особенности

- **Поле Password.** В этом поле хранится хешированный (зашифрованный) пароль, поэтому оно обычно оставляется без изменений

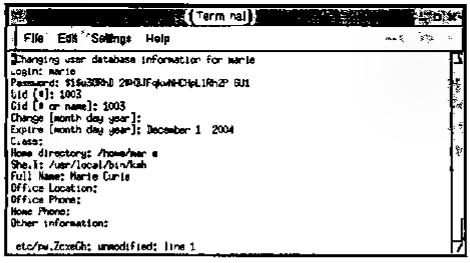


Рис 10 2 Команда `chpass` *позволяет изменять параметры учетной записи*

- **Поля `Change` и `Expire`** В этих полях записана дата когда пользователь должен сменить пароль, и дата окончания действия учетной записи При изменении пароля значение первого поля сбрасывается
- **Поля `Full Name` — `Other information`** Эта информация группируется и помещается в поле `Full name` файла `/etc/passwd` (в качестве разделителя используется запятая). Некоторые утилиты распознают это поле и отображают его содержимое подобно команде `chpass` Другие утилиты игнорируют все кроме имени пользователя.

Файл /etc/master.passwd

Суперпользователь может отредактировать файл `/etc/master.passwd`, а затем распространить изменения на другие файлы с помощью команды `pwd_mkdb`. Файл `/etc/master.passwd` содержит набор записей по одной для каждого пользователя Вот пример

```
marie:$1$u3ORhD12$HBJFqkwNHCHpL1Rh2P 6U1:1003:1003 .0:1101877200 ↵
Marie Curie:/home/marie:/usr/local/bin/ksh
```

Каждая запись состоит из нескольких полей разделенных двоеточиями Назначение полей таково.

- **Имя пользователя** В первом поле (`marie`) записано регистрационное имя пользователя
- **Пароль.** Во втором поле находится зашифрованный пароль. В файле `/etc/passwd` в этом поле стоит звездочка (`*`), указывающая на то, что фактический пароль хранится в файле `/etc/master.passwd` Если же звездочка встречается в файле `/etc/master.passwd`, значит, учетная запись не предназначена для регистрации пользователя в системе Попытка войти в систему приведет к выдаче сообщения об ошибке Пустое поле означает отсутствие пароля. Такая конфигурация опасна, но если, к примеру, забыт пароль суперпользователя, можно временно очистить это поле, а затем войти в систему и ввести новый пароль
- **Идентификатор пользователя** В третьем поле (1003) хранится значение UID связанное с учетной записью

- **Идентификатор группы.** В четвертом поле (1003) хранится идентификатор основной группы. Если пользователь входит в другие группы, то информация о них находится в файле `/etc/passwd`.
- **Класс.** Пятое поле предназначено для общей классификации пользователей и не используется стандартными утилитами FreeBSD. В файле `/etc/passwd` этого поля нет.
- **Дата изменения пароля.** В шестом поле (0) задается дата смены пароля, выраженная в секундах, прошедших с 1 января 1970 г. В файле `/etc/passwd` этого поля нет.
- **Срок действия учетной записи.** В седьмом поле (1101877200) задается срок действия учетной записи, выраженный в секундах, прошедших с 1 января 1970 г. В файле `/etc/passwd` этого поля нет.
- **Полное имя и сведения о пользователе.** В восьмом поле (`Marie Curie`) хранится полное имя пользователя. Разрешается указывать через запятую служебный адрес, рабочий и домашний телефоны, а также другую информацию.
- **Начальный каталог.** В девятом поле (`/home/marie`) задается начальный каталог пользователя.
- **Стандартный интерпретатор команд.** В последнем поле (`/bin/ksh`) задается стандартный интерпретатор команд пользователя.

Все эти поля разрешается редактировать, хотя удобнее пользоваться командами `chpass` и `passwd`. Любая ошибка в файле может привести к тому, что учетная запись станет недоступной. Файл `/etc/master.passwd` обычно просматривают, чтобы удостовериться в целостности базы данных пользователей. Если в файле присутствуют записи, которых не должно быть, или для учетных записей, не являющихся регистрационными, задан пароль, то система наверняка была взломана.

После внесения изменений в файл `/etc/master.passwd` нужно распространить их на другие конфигурационные файлы. Для этого предназначена команда `pwd_mkdb`.

```
# pwd_mkdb -p /etc/master.passwd
```

Проверяйте даты создания файлов `/etc/passwd`, `/etc/master.passwd`, `/etc/pwd.db` и `/etc/spwd.db`. Если текстовые версии файла паролей (`passwd` и `master.passwd`) созданы позднее, чем бинарные (`pwd.db` и `spwd.db`), то в них могут присутствовать неутвержденные изменения. Это может также свидетельствовать о взломе системы: злоумышленник восстановил версии файлов `passwd` и `master.passwd`, которые существовали до взлома, но файлы баз данных содержат обновленную информацию, поэтому взломанные учетные записи доступны.

Изменение идентификаторов пользователей

Сложнее всего изменить идентификатор пользователя, связанный с учетной записью. Это может понадобиться для согласования идентификаторов в системах, получающих совместный доступ к файлам посредством NFS или установленных на одном компьютере. Технически такое изменение легко осуществить, выполнив команду `chpass` либо отредактировав файл `/etc/master.passwd` и выполнив команду `pwd mkdb`. Но нужно помнить о двух возможных проблемах.

- **Идентификаторы существующих файлов не должны изменяться.** Поскольку FreeBSD хранит идентификатор владельца файла, смена идентификатора учетной записи приведет к тому, что существующие файлы пользователя внезапно

перейдут к неизвестному владельцу Чтобы устранить проблему, найдите с по мощью команды `find` такие файлы как описано на с. 250, и выполните для них команду `chown`. При работе с начальным каталогом пользователя удобно применять опцию `-R` команды `chown`

- **Если пользователь зарегистрирован в системе, он может вести себя непредска зуемо** Если на момент изменения пользователь зарегистрирован в системе, возникнет неразбериха. В каждом сеансе доступа выполняется ряд процессов, и их идентификатор владельца не изменится. Такие процессы могут потерять доступ к файлам пользователя, с которыми они работали, поэтому лучше по просить пользователя временно выйти из системы

Обычно нет причин изменять идентификатор пользователя существующей учетной записи При правильном планировании учетные записи пользователя в разных систе мах должны с самого начала иметь одинаковые идентификаторы

Удаление учетных записей

Иногда учетные записи приходится удалять Служащие меняют работу студенты оканчивают вуз или переходят на другой курс и т.д. На первый взгляд это не кажется важным. Какой может быть вред от неактивной учетной записи? Однако причины для удаления есть Одна из них — место на диске, занимаемое пользовательскими файла ми В крупной системе даже идентификатор пользователя (а возможно, и идентифи катор группы) является важным ресурсом. Но самое главное, неиспользуемые учетные записи представляют собой удобные объекты для взлома и стартовую площадку для последующих злонамеренных действий, поэтому их существование несет угрозу безо пасности системы. Во FreeBSD есть средства удаления таких записей Необходимо также выискать файлы, оставшиеся от прежних пользователей.

Команда rmuser

Основное средство удаления учетных записей — команда `rmuser` Ее синтаксис таков

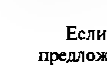
```
rmuser [-y] [пользователь]
```

Команда `rmuser` выполняет следующие действия:

- удаляет задания демона `cron`, запланированные пользователем,
- удаляет задания команды `at`, запланированные пользователем (см главу 6)
- уничтожает процессы запущенные пользователем
- удаляет относящиеся к пользователю записи из файлов паролей (`/etc/passwd` `/etc/master.passwd`, `/etc/pwd.db` и `/etc/spwd.db`);
- спрашивает, нужно ли удалить начальный каталог пользователя и в случае положительного ответа удаляет все дерево каталогов,
- удаляет почтовую очередь пользователя из каталога `/var/mail`
- удаляет файлы пользователя из каталогов `/tmp` `/var/tmp` и `/var/tmp/vi recover`,
- удаляет учетную запись пользователя из всех групп в файле `/etc/group` и саму группу, если пользователь является ее единственным членом

Короче говоря, команда `rmuser` отменяет все действия, выполнявшиеся при соз дании учетной записи, а также удаляет все файлы, создаваемые автоматически в про цессе работы с учетной записью. Во многих случаях больше ничего и не требуется

По умолчанию команда rmuser отображает соответствующую запись файла /etc/master.passwd и спрашивает нужно ли ее удалить Это необходимая мера предосторожности, ведь в файле указано полное имя пользователя что позволяет удостовериться в правильном выборе Но если задать опцию y команда будет "предполагать", что пользователь утвердительно отвечает на все вопросы



Прежде чем удалять начальный каталог, создайте его резервную копию Впоследствии это позволит восстановить учетную запись, если окажется, что ее не нужно было удалять или понадобятся файлы из нее

Если в строке вызова комаиды rmuser не указать имя пользователя, команда предложит его ввести

Удаление записей из файла /etc/master.passwd

В некоторых случаях требуется удалить учетную запись, но не нужно выполнять все действия, предусматриваемые командой rmuser. Например может понадобиться сохранить группу, в которую входил пользователь, или выполнить запланированные им задания В таком случае следует вручную сделать все необходимое Самое главное — удалить учетную запись из файлов паролей. Для этого отредактируйте файл /etc/master.passwd и воспользуйтесь командой pwd mkdb

```
# pwd_mkdb -p /etc/master.passwd
```

Команда воссоздаст файлы паролей без устаревшей учетной записи, но в системе останутся файлы с прежним идентификатором пользователя, а это чревато проблемами в будущем. Если повторно задействовать этот идентификатор новый пользователь может, к примеру, обнаружнть почтовые сообщения прежнего пользователя содержащие конфиденциальную информацию

Удаление ничейных файлов

Команда rmuser пытается удалить все файлы, принадлежащие пользователю но это ей не всегда удается Проблемы возникают, когда в системе есть каталоги совместного доступа, в которых пользователи хранят свои файлы Такие файлы можно найтн с помощью команды find, которая описывалась в главе 8 В частностн, следует указать опцию -user *пользователь*, где аргумент опции — это имя (если команда вызывается до удаления пользователя) или ндентификатор пользователя Рассмотрим следующую команду:

```
# find /home -user 1003
```

Эта команда находит в каталоге /home все файлы, принадлежащие пользователю с ндентификатором 1003. Их можно удалить или переназначить другому владельцу с помощью команды chown Файлы, владелец которых отсутствует в системе, по прежнему можно использовать, но это зависит от прав доступа

В показанном примере просматривался только каталог /home, так как это наиболее вероятное местонахождение "ничейных" файлов. Если конфигурация системы допускает запись в другие каталоги, включите их в путь поиска или просканируйте все дерево каталогов

Точно так же ищут файлы принадлежащие приватной группе пользователя Вот эквивалентная команда find:

```
# find /home -group 1003
```

Работа с группами

Большая часть главы посвящена созданию пользовательских учетных записей и управлению ими однако учетные записи групп тоже играют важную роль Во FreeBSD не так много средств работы с ними поскольку эти учетные записи имеют гораздо меньше параметров Как правило, группа создается с помощью программы `sysinstall` или путем редактирования файла `/etc/group`

На заметку Как указывалось ранее, утилита `adduser` автоматически создает для каждого пользователя приватную группу

Зачем создавать новые группы? Если, например несколько пользователей работают над общим проектом, можно включить их в общую группу, обеспечив групповой доступ к файлам и каталогам проекта Часто создают группы привилегированных пользователей, которым разрешен доступ к определенным ресурсам или программам В следующем разделе мы подробнее поговорим о том, какие виды групп бывают и в чем их преимущества и недостатки

Создание групп с помощью программы sysinstall

На с 241 рассказывалось о том, как применять программу `sysinstall` для создания учетных записей. В случае групп процедура почти такая же Отличия начинаются с этапа 4 Вместо пункта меню `User` нужно выбрать пункт `Group`. Появится форма `User and Group Management` (рис 10 3). Задайте название группы (поле `Group name`), ее идентификатор (поле `GID`) и список пользователей (поле `Group members`)

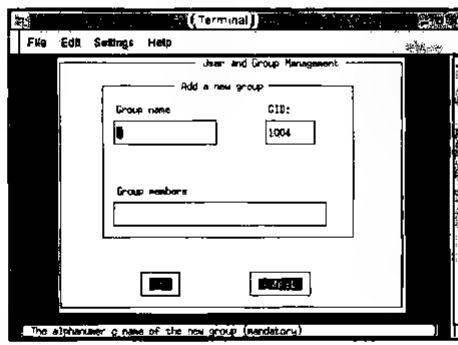


Рис 10.3 Программа `sysinstall` позволяет создавать новые группы

Совет Если создаются учетные записи пользователей входящих в новую группу, создайте сначала саму группу, а затем укажите ее в качестве ствндартной группы новых пользователей

Редактирование файла /etc/group

Можно создать группу, непосредственно отредактировав файл `/etc/group`. Подобно файлам `/etc/passwd` и `/etc/master.passwd`, он состоит из набора записей, поля которых разделяются двоеточиями. Вот пример записи

```
users:*:1001:marie,charles
```

Она содержит четыре поля.

- **Название группы**. В первом поле указывается название группы (`users`).
- **Пароль**. Групповые пароли используются редко, поэтому во втором поле обычно стоит звездочка, свидетельствующая об отсутствии пароля.
- **Идентификатор группы**. В третьем поле задан идентификатор группы (`1001`).
- **Список членов**. Последнее поле представляет собой разделенный запятыми список членов группы. Обычно здесь указываются пользователи, для которых это не основная группа. Например, если в учетной записи пользователя `paul` группа `users` определена как стандартная, то в список членов этой группы в файле `/etc/group` пользователь `paul` не включается. По этой причине четвертое поле часто оказывается пустым, хотя сама группа таковой не является.

Безопасность учетных записей

Учетные записи и группы являются одними из основных объектов контроля во FreeBSD. Безопасность системы во многом зависит от того, какие группы существуют, какие пользователи в них входят, каковы стандартные права доступа к файлам и какой доступ разрешен к ключевым системным каталогам. Однако пользователи вправе самостоятельно определять атрибуты своих файлов, поэтому полный контроль над ними невозможен. Следует обучать пользователей правилам безопасной работы в системе.

Выбор типа группы

Ранее уже говорилось о том, что утилита `adduser` автоматически создает специальную группу для каждого нового пользователя. Такая группа называется *приватной*, поскольку обычно в нее входит единственный пользователь. Другой подход заключается в создании групп под конкретные проекты или групп, объединяющих пользователей по какому-либо признаку. У каждого подхода есть преимущества и недостатки.

Приватные группы

Приватная группа существует для каждого пользователя. На первый взгляд это кажется бессмысленным, поскольку права владельца и группы становятся избыточными. Но не забывайте, что это только установки по умолчанию. В таком варианте все пользователи системы изолированы друг от друга с точки зрения безопасности, и права доступа назначаются либо владельцу, либо всем остальным пользователям. Но ничто не запрещает владельцу файла добавить в свою группу другого пользователя. В этом и заключается основное достоинство приватных групп. Владелец файлов самостоятельно определяет, кому разрешить доступ к ним. Предположим, в системе работают пользователи `marie`, `paul`, `charles` и `jane`, у каждого из которых есть приватная группа. Если Мэри хочет предоставить Полу доступ к своим файлам, то добавляет пользователя `paul` в группу `marie`. Какие конкретно права получит Пол, зависит от нее.

Стратегия приватных групп позволяет пользователям самостоятельно вести списки контроля доступа. Если Мэри разрешает доступ только Полу а Пол разрешает доступ Мэри и Чарльзу, то это легко реализовать — достаточно отредактировать списки членов групп `marie` и `paul`

Недостатком этого подхода является то, что пользователи не могут назначать специфические атрибуты доступа для разных групп файлов Чтобы обойти это ограничение нужно создать несколько приватных групп для разных целей

Группы совместных проектов

Другой подход заключается в создании групп, которые логически не связаны с одним конкретным пользователем-владельцем, но охватывают некоторое количество пользователей. Например, в системе могут существовать группы `physics` и `biology` первая из которых объединяет пользователей `marie` и `paul`, а вторая — пользователей `charles` и `jane`. Если пользователь входит сразу в несколько групп, он волен выбирать, к какой группе отнести тот или иной файл и задавать соответствующие права доступа.

Группы совместных проектов удобны в тех случаях когда существует четкое распределение обязанностей между пользователями системы и члены каждой такой группы имеют одинаковый доступ к файлам проекта. Если же права некоторых членов группы нужно как-то ограничить возникнут сложности и придется создавать иерархию групп

Смешанные группы

Чаще реализуется смешанный подход к созданию групп, когда у каждого пользователя есть приватная группа и, кроме того, в системе существуют группы совместных проектов. Это позволяет более гибко определять права доступа. Вообще говоря, разделение на два типа групп очень условно Все зависит от точки зрения Группу можно рассматривать как средство контроля доступа к файлам конкретного пользователя и как логическое объединение пользователей

Задание стандартных праа доступа к файлам

Стратегия безопасности системы диктует выбор стандартных прав доступа к создаваемым файлам. Это осуществляется с помощью команды `umask`, которая задает одноименную характеристику учетной записи (см. главу 8) В качестве аргументов команды указывается восьмеричное число, определяющее биты доступа, которые не должны устанавливаться по умолчанию

```
$ umask 027
```

В этом примере посторонним пользователям запрещается любой доступ к создаваемым файлам, а членам группы запрещается запись в файлы (FreeBSD сбрасывает биты выполнения для обычных файлов независимо от значения `umask`) Команду `umask` удобно вызывать в общесистемных сценариях конфигурации, чтобы одна установка распространялась на всех пользователей. Но помните, что пользователи могут переопределить эту установку в локальных конфигурационных файлах.

По умолчанию значение `umask` выбирается равным 022 Для файлов это означает режим доступа 644 (`-rw-r--r--`), а для каталогов 755 (`drwxr xr-x`). Во многих случаях такая конфигурация вполне соответствует потребностям системы. Все пользователи получают право читать чужие файлы, но не могут редактировать и удалять их В целях повышения безопасности можно устанавливать маску 027 и даже 077 чтобы по умолчанию доступ к файлам имели только владельцы В открытой среде подойдет маска 002 которая разрешает групповую запись в файлы Эта конфигурация не пред

ставляет особой угрозы если применяется стратегия приватных групп Маска 000 не налагает никаких ограничений

Значение `umask` влияет на атрибуты новых файлов создаваемых пользователями. Системные файлы сохраняют права доступа, полученные в процессе инсталляции Большинство файлов FreeBSD принадлежит пользователю `root` и группе `wheel` и имеет режим доступа 644 (`rw-r--r-`) либо 755 (`rwxr-xr-x`) Менять их как правило не приходится

Резюме

Во FreeBSD учетные записи создаются не только для удобства пользователей, но и для обеспечения безопасности системы. В однопользовательской системе единственную учетную запись можно создать непосредственно в процессе инсталляции FreeBSD. В случае же многопользовательской системы управление учетными записями является важным элементом системного администрирования. Учетные записи создаются с помощью программ `sysinstall` и `adduser`. Утилиты `passwd` и `chpass` позволяют модифицировать параметры учетных записей, а команда `rmuser` удаляет учетную запись Эти же действия можно выполнять путем непосредственного редактирования конфигурационных файлов. Схожим образом осуществляется управление группами пользователей При работе с учетными записями и группами следует обязательно помнить о безопасности системы, устанавливая их характеристики в соответствии с правилами разграничения доступа, принятыми в организации

Полный справочник по

Глава 11

Инсталляция программного обеспечения

В процессе инсталляции FreeBSD вместе с системой устанавливается целый ряд программных пакетов. Но, конечно, рано или поздно придется добавлять новое программное обеспечение (ПО). Умение найти и инсталлировать нужную программу важно для любого системного администратора. Существует несколько источников программного обеспечения для FreeBSD, включая официально поддерживаемые пакеты, полуофициально перенесенные во FreeBSD популярные пакеты, программы с открытым исходным кодом и коммерческие приложения.

Помимо инсталляции новых программ иногда приходится обновлять существующие. Здесь есть свои особенности, о которых следует знать (обновление всей операционной системы — тоже важная тема, она будет рассматриваться в главе 28). Мы также поговорим о степени доверия к источнику, из которого была получена программа. Большинство программ, доступных в Internet, поступает из надежных источников, но есть и такие, которых следует избегать.

Виды программных пакетов

Прежде чем говорить об инсталляции, необходимо понять, в каком виде может распространяться программное обеспечение. Это может быть либо исходный код, либо готовый к использованию исполняемый код. Важен также способ распространения. Иногда программа, распространяемая определенным способом, доступна только в исходном коде или только в исполняемом виде.

Исходный и исполняемый код

Программисты пишут программы преимущественно на *языках высокого уровня*, таких как C, C++ или Perl. У этих языков есть много достоинств, в частности *переменные* (хранят данные, которые могут меняться в процессе выполнения программы) англоязычные имена функций и *библиотеки* (готовый код, решающий наиболее распространенные задачи программирования). Файлы, написанные на таком языке, называются файлами *исходного кода*. Их можно редактировать с помощью обычного текстового редактора, например `vi` или Emacs. Часто программа состоит из множества небольших файлов исходного кода.

Компьютеры работают с числами, поэтому необходимо установить соответствие между низкоуровневыми инструкциями центрального процессора и командами высокоуровневого языка. Один из способов сделать это — использовать *компилятор*, который транслирует исходный код в исполняемую форму. Исполняемый файл записан на машинном языке, понятном процессору. Компилятор создает исполняемый файл для конкретного процессора, поэтому такой файл может использоваться лишь в системах с процессором определенной модели или определенного семейства, например x86 или PowerPC.

Другой способ обработки исходного кода заключается в его непосредственной интерпретации с помощью программы, называемой *интерпретатором*. Сценарии, выполняемые интерпретатором, работают медленнее, чем те же сценарии, но скомпилированные. Интерпретируемые программы проще разрабатывать, к тому же такая программа может работать на различных процессорах. Язык программирования обычно реализуется либо компилятором, либо интерпретатором. Например, языки C и C++ являются компилируемыми, а Perl — интерпретируемым.

UNIX-системы традиционно функционируют на самых разных процессорах, поэтому программы для этой системы часто распространяются в исходном коде. Это позволяет системным администраторам компилировать программы на любом компьюте-

ре, по крайней мере теоретически Недостатком такого подхода является то, что на компиляцию уходит время Компиляция может занимать от нескольких секунд до нескольких часов, в зависимости от размера программы и скорости процессора. Кроме того, возможность скомпилировать программу в любой системе зачастую справедлива лишь в теории. На практике программу приходится модифицировать для определенных операционных систем или процессоров, и иногда это непростая задача. Программы, распространяемые в бинарном виде, проще инсталлировать, при условии, конечно, что они компилировались в системе с аналогичными или хотя бы сходными характеристиками.

Программное обеспечение для FreeBSD, как правило, доступно в обеих формах Предпочтительнее использовать готовые бинарные пакеты, если они есть Исходные коды нужны, когда требуется модифицировать программу или при инсталляции готовой программы возникают проблемы, которые можно устранить путем перекомпиляции (проблемы обычно связаны с тем, что остальное программное обеспечение системы слишком устарело или, наоборот, значительно новее, чем сама программа). Есть и программы, которые доступны лишь в одной из двух форм. В частности, некоторые свободно распространяемые пакеты не предлагаются скомпилированными для FreeBSD, и их приходится компилировать самостоятельно. Коммерческие приложения часто поставляются без исходного кода и инсталлируются в бинарном виде

Способы распространения

Способ распространения программы определяет, где ее следует искать а иногда и условия ее лицензирования Перечислим наиболее популярные способы

- **Пакеты FreeBSD** Это заранее скомпилированные исполняемые программы готовые для быстрои и относительно беспроблемной инсталляции, при условии, что пакет инсталлируется в корректной версии FreeBSD (пакет, предназначенный для одной версии, не всегда работает в другой) Такие пакеты находятся на инсталляционных носителях FreeBSD и доступны на различных FTP зеркалах
- **Переносимые приложения для FreeBSD** Коллекция переносимых приложений для FreeBSD представляет собой базу данных, в которой хранятся ссылки на программы вместе с информацией об их модификации во FreeBSD Эта информация позволяет скомпилировать программу, имея лишь ее исходный код и инсталлировать ее в системе. Преимуществом такого подхода является то, что система автоматически вносит изменения в исходный код. Коллекция переносимых приложений существует в виде файлов-указателей в каталоге `/usr/ports` Перед началом инсталляции программы FreeBSD загружает ее исходный код из Internet
- **Универсальное программное обеспечение для UNIX.** Некоторые UNIX программы с открытым исходным кодом не были официально перенесены во FreeBSD По крайней мере, они не существуют в виде FreeBSD-пакета или переносимого приложения. Такие программы можно компилировать и запускать во FreeBSD, но их использование связано с определенным риском, а кроме того, их сложнее администрировать Обычно они доступны на Web- или FTP-узле разработчика Об общих принципах компиляции программ будет рассказываться в главе 30
- **Коммерческое ПО** Производители коммерческих программ обычно распространяют свои продукты в бинарном виде Эти программы можно купить в компьютерном магазине или на Web-узле поставщика Иногда они инсталлируются как пакет или переносимое приложение, но чаще к ним прилагается программа-инсталлятор или файл с инструкциями для инсталляции вручную

При распространении программ через Internet есть возможность получить самую последнюю версию программы тогда как на компакт дисках чаще всего содержатся более ранние версии. Однако необходимо следить за актуальностью ссылок. К примеру, в каталоге /usr/ports могут храниться устаревшие ссылки на переносимые приложения, поэтому результат не всегда оправдывает ожидания.

Пакеты FreeBSD поставляются в формате, который удобен с точки зрения обслуживания системы. Утилиты создания пакетов добавляют информацию о пакетах в базу данных программ, установленных на компьютере. С помощью этой базы данных легко узнать версию имеющейся программы, деинсталлировать программу или обновить ее версию. Все это чрезвычайно важно для администраторов FreeBSD. Программы из коллекции переносимых приложений тоже инсталлируются как пакеты, так что преимущества пакетного формата распространяются и на них.

У пакетного формата есть еще одно достоинство: он позволяет задавать другие пакеты, от которых зависит используемый пакет (так называемые *зависимости*). Служебные утилиты могут использовать эту информацию, чтобы блокировать инсталляцию пакета, для которого не соблюдаются зависимости; чтобы автоматически загружать и инсталлировать зависимые пакеты, чтобы предотвращать деинсталляцию пакетов, от которых зависят другие пакеты. Это помогает сохранять целостность системы, защищая ее от случайного разрыва межпакетных зависимостей.

Поиск программного обеспечения для FreeBSD

В полном варианте FreeBSD занимает несколько компакт-дисков, на которых записано большое число программных пакетов, как говорится, "на все случаи жизни". Но этого далеко не всегда достаточно. В таких ситуациях важно знать, где найти нужную программу. Периодически следует также проверять наличие обновлений. В программах чуть ли не ежедневно выявляют различные "бреши" и ошибки, поэтому устаревшие пакеты угрожают безопасности системы.

Использование стандартного системного ПО

Наиболее важным источником программного обеспечения для FreeBSD является собственно системная коллекция пакетов. В нее входит большинство популярных программ. Многие из них представлены также в коллекции переносимых приложений.

Все эти пакеты доступны на FTP-узле FreeBSD и его "зеркалах", а также на инсталляционном носителе. Пакеты распространяются в виде tar-архивов с расширением .tgz. Такие файлы инсталлируются с помощью утилиты pkg_add, как будет описано далее. Если точное имя файла неизвестно, поищите его с помощью программы sysinstall.

На заметку

Основной FTP-узел FreeBSD имеет адрес ftp://ftp.freebsd.org. Чтобы снизить огромную нагрузку на него, по всему миру был организован целый ряд *зеркальных* серверов. На них хранятся те же самые файлы, и пользователь для ускорения загрузки может выбрать ближайший к себе узел. Список "зеркальных" серверов доступен по адресу http://www.freebsdmirrors.org/FBSDsites.php3.

Переносимые приложения для FreeBSD ищутся совсем по-другому. Если соответствующая коллекция инсталлирована в системе, то база данных со ссылками находится в каталоге /usr/ports. Он разделен на подкаталоги по типам программ, например /usr/ports/editors и /usr/ports/science. В каждом из них есть подкаталоги, посвященные конкретным программам, но в них содержатся не сами программы, по

крайней пере первоначально В этих каталогах приведена информация о том, где найти исходный код программы, там же находятся файлы "заплат" и инструкции по компиляции программы Чтобы инсталлировать такую программу, нужно ввести специальную команду, которая заставит FreeBSD загрузить программу скомпилировать и инсталлировать ее

Поиск ПО общего назначения для UNIX

Большое количество программного обеспечения для FreeBSD доступно в виде пакетов и переносимых приложений Но есть программы, для которых существует только исходный код Автор такой программы может скомпилировать ее для одной-двух своих любимых платформ, но среди них не обязательно будет FreeBSD

Если имя программы известно, ее можно поискать различными способами

- **Web-поиск**. Поисковые системы наподобие Google (`www.google.com`) являются хорошей отправной точкой. С их помощью проще всего искать программы с нестандартными именами
- **Sourceforge**. На Web-узле Sourceforge (`http://sourceforge.net`) приведена информация о многочисленных открытых проектах и программах с открытым исходным кодом Здесь же есть поисковый механизм позволяющий вести поиск по названию проекта или ключевому слову
- **Web-узлы других операционных систем**. Иногда ссылки на нужные программы удается найти на Web-узле другой операционной системы. Исполняемые файлы, подготовленные в другой системе, могут не работать во FreeBSD, поэтому ищите ссылки на Web-узел самой программы или на ее исходный код.

Те же самые ресурсы можно использовать для поиска программ определенного типа Например, если в поле поиска на Web-узле Sourceforge ввести `editor`, будет выдан список редакторов текстовых, звуковых и т д Естественно, чем точнее критерий поиска, тем полезнее будут полученные результаты

Поиск коммерческого ПО

Большинство программ для FreeBSD распространяется с открытым исходным кодом Они либо входят в состав самой операционной системы, либо доступны для загрузки на Web-узле разработчика Как правило, у каждой из них есть и коммерческие аналоги. Нельзя однозначно сказать, чему стоит отдавать предпочтение, — все зависит от конкретной задачи Иногда коммерческий продукт оказывается единственно возможным выбором Например, программа RealPlayer компании Real Networks (`www.rea..com`) является наиболее удобным средством воспроизведения мультимедийных файлов формата RealAudio.

Коммерческие приложения можно искать через поисковые системы или специализированные Web-узлы В первую очередь обратитесь на Web страницу поставщиков коммерческого ПО для FreeBSD по адресу `http://www.freebsd.org/commercial/software.html`

На заметку

Определение "коммерческая" еще не означает, что за программу нужно платить деньги. Некоторые коммерческие продукты доступны бесплатно Иногда это усеченные версии платных программ, а иногда условия лицензирования допускают бесплатное применение, например в учебных заведениях или в личных целях Есть компании, которые продают расширенные версии программ изначально распространяемых с открытым исходным кодом

Инсталляция пакетов

Пакеты — самый простой формат распространения ПО с точки зрения обработки особенно для администраторов-новичков Чтобы инсталлировать пакет нужно вы брать его имя в меню программы `sysinstall` или сначала загрузить пакетный файл а затем выполнить для него команду `pkg_add`. FreeBSD ведет базу данных с инфор мацией об инсталлированных пакетах поэтому легко узнать, инсталлирована ли про грамма и какова ее версия (когда инсталлируется программа из коллекции переноси мых приложении сведения о ней добавляются в эту же базу данных)

Использование программы sysinstall

Программа `sysinstall` уже неоднократно упоминалась в предыдущих главах книги Это универсальное средство системного администрирования одно из основных достоинств которого — возможность использования в качестве инсталлятора пакетов С помощью программы `sysinstall` можно найти нужный пакет (хранящийся на ком-пакт-диске или сетевом сервере) и установить его. Это особенно удобно, если известен тип программы, но не ее точное имя Ниже описана процедура инсталляции пакета

1. Перейдите в режим суперпользователя и введите **`/stand/sysinstall`** для запус ка программы `sysinstall`.
2. Выберите в основном меню программы пункт `Configure`, чтобы перейти в меню конфигурирования FreeBSD (`FreeBSD Configuration Menu`).
3. Выберите в меню конфигурирования пункт `Packages`, чтобы перейти в меню `Choose Installation Media`, где можно задать источник получения пакетов.
4. Выберите носитель, с которого будут инсталлироваться пакеты Варианты таковы `CD/DVD` (локальный дисковод CD-ROM или DVD-ROM), `FTP`, `FTP Passive` и `HTTP` (локальный или удаленный FTP-узел), `DOS` (локальный раздел FAT), `NFS` (локальный сервер NFS) и `File System` (локальный раздел FreeBSD). В первом из вариантов необходимо вставить компакт-диск в дисковод Во втором варианте программа отобразит список FTP-узлов, из которых необходнмо выбрать нужный узел В списке присутствует специальный элемент `URL`, предназначенный для за дания адреса узла, включая тот узел, на котором вы непосредственно работаете Далее система прочтет список доступных пакетов с выбранного источника. Это может занять несколько минут По окончании программа отобразит перечень ка тегорий пакетов (рис 11 1)

Совет Если в сети есть несколько FreeBSD-систем, можно скопировать со держимое FTP-узла FreeBSD на локальный FTP-сервер и использовать его для инсталляции и обслуживания остальных систем

На заметку Список пакетов будет отличаться в зависимости от источника В част ности, если выбрать пункт `CD/DVD`, в список будут включены только те пакеты, которые записаны на компакт-диск, вставленный в дисковод. В случае инсталляционного набора из нескольких компакт-дисков придется повторить эту процедуру несколько раз чтобы определить все доступные пакеты

5. Выберите категорию к которой относится инсталлируемал программа Многие пакеты входят сразу в несколько категорий. В частности категория `All` объеди няет все доступные пакеты Когда выбор сделан, система отображает список па кетов, входящих в эту категорию (рис 11 2).

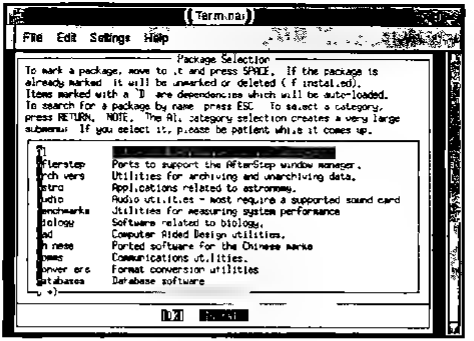


Рис 11 1 FreeBSD сортирует пакеты по категориям

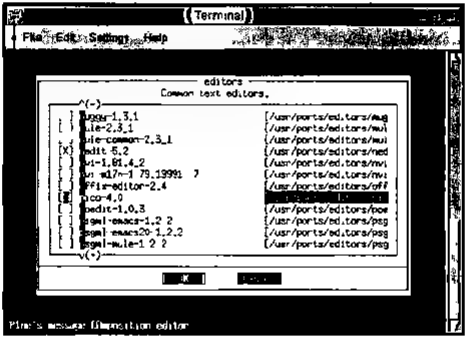


Рис 11 2 Для каждой категории отображается перечень доступных пакетов, а в нижней строке экрана приводится описание выбранного пакета

6 Некоторые пакеты уже выбраны, о чем свидетельствует метка X слева от названия пакета Чтобы выбрать пакет, переведите курсор на соответствующую строку и нажмите пробел.

7 По окончании нажмите <Tab>, чтобы перевести курсор на кнопку OK и нажмите <Enter> Снова появится меню выбора пакетов (рис 11 1)

8 Если нужно выбрать пакеты из других категорий, повторите пп 5 7

9 Нажимайте клавишу <Tab> до тех пор, пока курсор не окажется на кнопке Install, после чего нажмите <Enter> Программа sysinstall отобразит перечень инсталлируемых пакетов Выберите OK Далее программа загрузит файлы по сети (если это необходимо) и установит их в системе, отображая сведения о ходе

инсталляции каждого пакета. В конце снова появится меню FreeBSD Configuration Menu.

10. Выберите Cancel, а затем — Exit Install, чтобы выйти из программы.

Всегда полезно просматривать список пакетов, отображаемый программой sysinstall, так как в нем можно найти много полезных утилит. Обращайте внимание на нижнюю строку экрана, в которой отображается краткое описание пакета (рис. 11.2).

На заметку Как ни странно, единственный способ выйти из меню Package Selection — выбрать команду Install, даже если ни один пакет не выбран. В этом случае программа sysinstall сообщит об отсутствии пакетов для инсталляции, после чего появится меню FreeBSD Configuration Menu.

Использование специализированных утилит

Программу sysinstall удобно использовать для инсталляции пакетов, но она не лишена недостатков. Один из них заключается в том, что программа работает только с пакетами, которые доступны на инсталляционном носителе FreeBSD. Это может быть компакт-диск, DVD-диск или жесткий диск FTP-сервера (если есть доступ к жесткому диску сервера, можно добавить в файл INDEX ссылки на другие пакеты, но это лишь повышает сложность использования программы). Другая проблема — отсутствие информации о размере пакетов. После начала инсталляции программа sysinstall сообщает объем копируемых данных, однако этого недостаточно, чтобы оценить, стоит ли устанавливать крупный пакет.

Проблемы можно решить с помощью низкоуровневых утилит работы с пакетами. Эти утилиты находятся в каталоге /usr/sbin, и их имена начинаются с префикса pkg_. Наиболее важная из них — pkg_add, она добавляет заданный пакет в систему. Именно она вызывается программой sysinstall в процессе инсталляции пакетов.

Внимание! Команда pkg_add работает с пакетами, полученными из любого источника, не проверяя их надежность. Это особенно опасно, если источником пакета не является инсталляционный носитель FreeBSD. Прежде чем использовать команду, прочтите раздел "Проверка аутентичности програмы".

Синтаксис команды pkg_add таков:

```
pkg_add [-vInfrRMS] [-t шаблон] [-p префикс] имя_пакета[
```

Ниже описано назначение опций команды.

- -v — включает режим расширенных сообщений, который полезен при отладке.
- -I. В состав пакетов часто входят инсталляционные сценарии, осуществляющие настройку программы для конкретной системы. По умолчанию они выполняются. Эта опция отключает инсталляционные сценарии.
- -n — заставляет команду только сообщать о действиях, которые будут предприняты для инсталляции пакета, но не выполнять саму инсталляцию.
- -f. По умолчанию, прежде чем инсталлировать пакет, команда проверяет наличие зависимостей. Опция -f отключает проверку зависимостей. Обычно это нежелательно, если только зависимые программы не были инсталлированы отдельно от системы управления пакетами.

- `r` Обычно в последнем аргументе задается имя пакетного файла или URL-адрес. При наличии опции `-r` команда воспринимает последний аргумент как имя пакета, который следует найти в Internet.
- `R` — запрещает запись информации о пакете в базу данных. Эта опция почти никогда не используется.
- `-M` — включает управляющий режим, в котором команда извлекает содержимое пакета во временную область и направляет список файлов в стандартный выходной поток. Это позволяет писать сценарии, модифицирующие работу команды.
- `S` — включает подчиненный режим, в котором команда читает из стандартного входного потока список файлов, находящихся во временной области, и объединяет их в пакет. Это позволяет писать сценарии, модифицирующие работу команды.
- `t` *шаблон* — задает каталог, используемый для временного хранения файлов. По умолчанию это каталог `/var/tmp/instmp.XXXXXX` (вместо XXXXXX утилита подставляет уникальный идентификатор). Если раздел `/var` слишком мал для хранения временных файлов, задайте другой раздел. Имя каталога обязательно должно заканчиваться на XXXXXX, чтобы утилита могла заменить эти символы идентификатором.
- `p` *префикс* — задает каталог, в который команда будет инсталлировать содержимое пакета. Обычно нужный каталог указан в самом пакете, но опция `p` позволяет переопределить его.

Единственный обязательный аргумент команды — имя инсталлируемого пакета. Это может быть либо имя файла на локальном диске, либо URL-адрес файла на FTP-сервере. Рассмотрим следующие команды:

```
# pkg_add ftp://ftp3.freebsd.org/pub/FreeBSD/ports/i386/packages-
5-current/editors/zile-1.6.1.tgz
# pkg_add zile-1.6.1.tgz
```

Совет Прежде чем инсталлировать пакет, проверьте с помощью команды `pkg_info`, нет ли в системе его предыдущей версии. Если это так, лучше обновить пакет, воспользовавшись командой `pkg_update`, чем инсталлировать его заново.

Первая из приведенных команд инсталлирует редактор `zile` непосредственно с FTP-узла `ftp3.freebsd.org`. Вторая команда устанавливает тот же пакет из текущего каталога локального диска. Если есть инсталляционный компакт-диск FreeBSD, то для установки пакета нужно смонтировать компакт-диск и указать утилите полный путь к файлу.

На заметку Команда `pkg_add` работает не со всеми `tar`-архивами. Пакеты FreeBSD должны содержать специальные файлы, которые используются командой для добавления информации о пакетах в базу данных.

Запрос информации о пакете

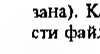
Команда `pkg_info` позволяет получить информацию о пакете. Эта команда принимает большое число опций, заставляющих ее отображать различные сведения о пакете или группе пакетов. Все они описаны на соответствующей man-странице. Как правило, команда используется одним из следующих способов:

- **Запрос ко всей базе данных** Если ввести `pkg_info`, будут выданы имена всех инсталлированных пакетов с кратким описанием каждого из них
- **Запрос к инсталлированному пакету** Если задать имя пакета (включая номер версии), команда выдаст его многострочное описание. Например, команда `pkg_info nedit-5.2` отображает описание пакета NEdit 5.2, при условии что он инсталлирован на компьютере
- **Поиск инсталлированного пакета.** Если точное имя пакета неизвестно воспользуйтесь опцией `-x`. Например, команда `pkg_info -x nedit` отображает информацию о пакете NEdit несмотря на отсутствие номера версии
- **Запрос к неинсталлированному пакету** Разрешается вместо имени инсталлированного пакета ввести имя пакетного *файла*, чтобы узнать его содержимое независимо от того, инсталлирован пакет или нет Например, с помощью команды `pkg_info nedit 5.2.tgz` можно просмотреть содержимое архива `nedit 5.2.tgz` перед его инсталляцией
- **Поиск файлов в пакете** Если нужно узнать какие файлы входят в пакет воспользуйтесь опцией `-L`. Например, команда `pkg_info L nedit-5.2` отображает список файлов пакета `nedit-5.2`
- **Поиск пакета для файла** Если нужно узнать имя пакета, к которому принадлежит заданный файл, укажите опцию `-W` Например, команда `pkg_info W /usr/X11R6/bin/nedit` ищет в базе данных имя пакета, из которого был инсталлирован файл `/usr/X11R6/bin/nedit`.

Команду `pkg info` удобно использовать как до, так и после инсталляции пакета Она позволяет узнать имя пакета, связанного с подозрительной или неправильно работающей программой, определить, куда будет записано его содержимое (и тем самым предупредить проблемы, связанные с возможными конфликтами или нехваткой места на диске), выяснить его назначение и т д. Рассмотрим следующую цепочку команд

```
$ pkg_info -W /usr/X11R6/bin/nedit
/usr/X11R6/bin/nedit was installed by package nedit 5 2
$ pkg_info -x nedit-5.2
```

Вторая команда выдает расширенную информацию о пакете `nedit-5.2` (не показана). Как видите, команду `pkg info` можно применять для выяснения принадлежности файлов, назначение которых непонятно



Если отчет, получаемый при указании опции x, не позволяет прояснить назначение программы, воспользуйтесь опцией `-L`, чтобы узнать, какие файлы были инсталлированы, и поищите среди них файлы документации. Такие файлы обычно хранятся в каталоге `/usr/share/doc` или `/usr/local/share/doc` либо в каталогах man-страниц

Коллекция переносимых приложений для FreeBSD

Пакеты FreeBSD удобны и просты в инсталляции Но не всегда можно найти заранее скомпилированный пакет или же пакет, подготовленный для имеющейся версии FreeBSD. Иногда перед инсталляцией программы необходимо внести изменения в ее исходный код. В подобных ситуациях прибегают к аьтернативным способам распространения программного обеспечения. Один из вариантов — предоставлять ис-

ходный код, который должен модифицироваться вручную с учетом особенностей системы. Это довольно гибкий подход, но он не позволяет воспользоваться преимуществами, которые обеспечивает система управления пакетами Другой вариант — использовать коллекцию переносимых приложений Существуют переносимые версии многих программ, обеспечивающие их компиляцию во FreeBSD По окончании компиляции система преобразует программу в пакет, что дает возможность применять к ней различные утилиты обработки пакетов, в частности `pkg info`. Недостатком такого подхода является увеличение времени, затрачиваемого на инсталляцию. Это особенно заметно, когда у программы множество неинсталлированных зависимостей В конечном счете, переносимые приложения являются промежуточным звеном между пакетами и обычным исходным кодом

Под переносом понимается процесс преобразования программы для работы на другой платформе. Перенос программ во FreeBSD имеет свои особенности Здесь к исходному коду программы добавляется набор "заплат", в которых описаны изменения, позволяющие корректно скомпилировать исходный код во FreeBSD. После того как разработчик написал программу и "заплаты" для нее, системный администратор FreeBSD может инсталлировать ее в автоматическом режиме

Чтобы получить дополнительные сведения о принципах разработки переносимых приложений, введите команду **`man ports`**.

Инсталляция и обновление коллекции

Коллекцию переносимых приложений можно инсталлировать непосредственно в ходе установки системы. Если это было сделано, в системе появится каталог `/usr/ports`, содержащий файлы указатели В противном случае каталог либо отсутствует, либо пуст, и коллекцию нужно инсталлировать вручную. Кроме того, ее необходимо периодически обновлять, особенно если планируется использовать ее для обновления версий программ.

Проще всего инсталлировать коллекцию с помощью программы `sysinstall` Вот как это делается.

1. Перейдите в режим суперпользователя и введите **`/stand/sysinstall`** чтобы запустить программу `sysinstall`
2. Выберите в основном меню программы пункт `Configure`, чтобы перейти в меню конфигурирования FreeBSD (`FreeBSD Configuration Menu`)
3. Выберите в меню конфигурирования пункт `Distributions` чтобы перейти в меню дистрибутивов.
4. Выберите дистрибутив `Ports` переведите курсор на кнопку `OK` и нажмите <Enter> Программа спросит, на каком носителе находится дистрибутив
5. Выберите носитель, как описывалось на с 260 Программа загрузит коллекцию переносимых приложений и инсталлирует ее. После этого можно выходить из программы

С этого момента в каталоге `/usr/ports` появятся подкаталоги, соответствующие категориям программ, например `/usr/ports/x11` и `/usr/ports/editors` В этих подкаталогах будут присутствовать каталоги самих программ

Особенностью коллекции является то, что файлы коллекции ссылаются на конкретные версии программ Таким образом, если программа обновлена после установки коллекции на жесткий диск, система будет автоматически инсталлировать устаревшую версию программы А если вдруг поменяется адрес FTP-узла программы, по-

пытка инсталляции закончится неудачей. Даже если в системе установлена самая последняя версия коллекции, это не означает, что ей соответствуют новейшие версии программ: возможно, разработчики просто не успели перенести их во FreeBSD. Чтобы свести проблемы к минимуму, следует периодически обновлять коллекцию с помощью CVS (Concurrent Versions System — система параллельного управления версиями). Ниже приведена необходимая для этого последовательность действий:

1. Если это еще не сделано, инсталлируйте пакет `cvsup-without-gui` с помощью утилит управления пакетами или средствами самой коллекции (во втором случае придется инсталлировать компилятор Modula-2 и его многочисленные зависимости — на это уходит много времени и дискового пространства).
2. Введите `cd /usr/ports/net/cvsupit`.
3. Введите **`make; make install`**. Эта команда установит и сконфигурирует ряд необходимых утилит, после чего запустит программу обновления исходных кодов системы (рис. 11.3).

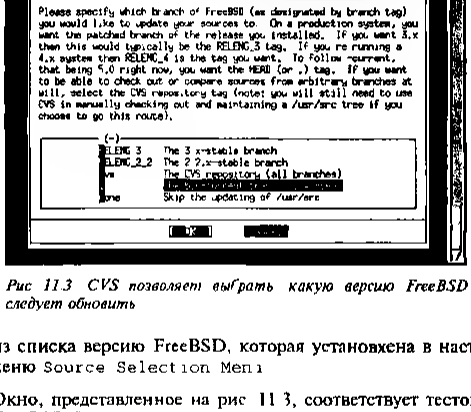


Рис. 11.3. CVS позволяет выбрать, какую версию FreeBSD следует обновить

4. Выберите из списка версию FreeBSD, которая установлена в настоящий момент. Появится меню `Source Selection Menu`.

На заметку Окно, представленное на рис. 11.3, соответствует тестовой бета версии FreeBSD 5.0. В окончательной версии системы должен появиться пункт `RELENG_5`.

5. В меню `Source Selection Menu` не выбирайте никаких пакетов. Просто нажмите <Tab>, а затем <Enter>, чтобы пропустить данный этап (пакеты, которые можно здесь выбрать, не имеют отношения к коллекции переносимых приложений).
6. Программа спросит, нужно ли проверить версию коллекции переносимых приложений. Выберите `Yes`.
7. Программа спросит, нужно ли проверить версию документации к FreeBSD. Лучше ответить `No`, чтобы ускорить процесс.
8. Программа попросит ввести имя базового каталога. Оставьте значение по умолчанию (`/usr`).

9. Программа выдаст список серверов CVSup. Выберите ближайший сервер.
10. Программа спросит, нужно ли выполнить обновление сейчас. Ответьте `No`, чтобы вернуться в режим командной строки.
11. Отредактируйте файл `/etc/cvsupfile`. Это конфигурационный файл сервера CVSup. Превратите в комментарий все строки, начинающиеся с `src-` (поставьте в начале строки символ '#'). В них заданы исходные файлы операционной системы, которые обновлять не нужно.
12. Введите следующую команду, чтобы обновить коллекцию переносимых приложений:

```
# /usr/local/bin/cvsup -g -L 2 /etc/cvsupfile
```

В процессе обновления система будет сообщать имена загружаемых файлов. Обычно обновление происходит очень быстро. Если же изменения существенны, оно длится дольше. Дольше всего обновляется файл `ports/INDEX`, `INDEX`, содержащий информацию обо всех переносимых приложениях системы. Если коллекция слишком устарела или сетевое соединение очень медленное, обновление может занять больше часа.

С этого момента в каталоге коллекции будут храниться ссылки на обновленные версии программ. В то же время программы, инсталлированные ранее, останутся в системе. Их версии не обновятся. На с. 269 будет рассказано, как обновить уже инсталлированные программы.

Совет

На Web-узле Fresh Ports (`www.freshports.org`) приведена информация о том, какие изменения вносятся в коллекцию переносимых приложений. Периодически посещайте этот Web-узел, чтобы узнать, не появились ли важные изменения и стоит ли загружать обновления.

Компиляция и инсталляция переносимого приложения

В каталоге `/usr/ports` содержатся подкаталоги, соответствующие категориям программ, примерно как в меню `Package Selection` программы `sysinstall` (рис. 11.1). Чтобы инсталлировать приложение из коллекции, выполните перечисленные ниже действия.

1. Введите **`pkg_info -x`** *имя пакета*, чтобы узнать, существует ли в системе пакет с таким именем. Если он есть, то, возможно, ничего инсталлировать не придется. Если требуется обновить существующий пакет, перейдите к разделу "Обновление программы".
2. Найдите каталог искомой программы. Это можно сделать с помощью команды `find` или `grep` (см. главу 8).
3. Перейдите в каталог программы.
4. В режиме суперпользователя введите **`make`**. Это заставит систему загрузить исходный код программы, применить к нему "заплаты", имеющиеся в подкаталоге `file`, и скомпилировать программу. Кроме того, система автоматически загрузит, скомпилирует и инсталлирует другие переносимые приложения, если программа зависит от них.
5. В режиме суперпользователя введите **`make install`**, чтобы инсталлировать программу. Информация о ней будет добавлена в пакетную базу данных. Этапы 4 и 5 можно объединить, воспользовавшись командой `make install`, но лучше проверить сообщения, выдаваемые по окончании компиляции, прежде чем приступать к инсталляции программы.

Перед началом инсталляции нужно убедиться в наличии средств разработки. Пакет GCC (GNU Compiler Collection) является стандартным компонентом FreeBSD и должен присутствовать в системе, но могут понадобиться служебные библиотеки или другие компиляторы, особенно для графических приложений. В каждом каталоге коллекции есть файл `Makefile`, в котором содержится URL-адрес основного Web-узла программы, поэтому можно проверить, какие средства разработки используются. Необходимые пакеты содержатся в разделе `devel` меню `Package Selection` программы `sysinstall`. Если не инсталлировать средства разработки заранее, система установит их сама, но это займет гораздо больше времени.

Теперь в соответствующем каталоге коллекции будет содержаться исходный код инсталлированного приложения. В некоторых случаях это удобно, так как исходный код можно модифицировать, чтобы исправить ошибки или поменять работу программы. Но он занимает место на диске, как и промежуточные файлы (*объектный код*), сгенерированные компилятором в процессе создания исполняемого файла. Чтобы удалить все лишнее, введите **`make clean`** в каталоге программы. Эта же команда в каталоге `/usr/ports` удалит "хлам", оставшийся после инсталляции переносимых приложений. Чтобы освободить дополнительное место на диске, удалите `tar`-архивы из каталога `/usr/ports/distfiles`. В нем находятся исходные файлы, загруженные с Web- или FTP-узлов программ. Их наличие позволяет системе избежать повторной загрузки программы в случае перекомпиляции.

Если в сети есть несколько FreeBSD-систем, то особый интерес представляет команда `make package`. Она создает пакет на основе переносимого приложения и устанавливает его в локальной системе, если само приложение еще не инсталлировано. Этот пакет можно затем скопировать в другие системы и инсталлировать с помощью команды `pkg_add`, не компилируя много раз.

Удаление и обновление ПО

Инсталляция программ — лишь один аспект работы администратора. Программы приходится также удалять и обновлять. Удаление неиспользуемых программ необходимо в целях экономии места на диске и полезно с точки зрения безопасности, особенно в отношении серверов и программ с установленным битом SUID. Что касается новых версий программ, то в них устраняются ошибки и расширяются функциональные возможности предыдущих версий. Некоторые ошибки напрямую связаны с безопасностью системы, поэтому своевременное обновление является важным элементом стратегии защиты.

Удаление программы

Если программа инсталлировалась в виде пакета, то удалить ее несложно. Для этого предназначена команда `pkg_delete`, имеющая следующий синтаксис:

```
pkg_delete [-dDfGinrvx] [-p префикс] имя_пакета[...]
```

Некоторые опции команды, в частности `-v`, `-f`, `-n` и `-p`, соответствуют одноименным опциям команды `pkg_add`. Назначение остальных опций таково:

- **`-d`**. По умолчанию команда не удаляет каталоги, если они не являются частью пакета. Опция `-d` заставляет команду удалять каталоги, которые стали пустыми в ходе деинсталляции пакета.

- `D` Пакеты иногда включают в себя сценарии деинсталляции Эти сценарии могут удалять учетные записи пользователей соответствующего сервера, оста навливать сервер, удалять конфигурационные файлы, не являющиеся офици альными компонентами пакета и т д Опция `-D` заставляет команду игнориро вать эти сценарии.
- `G` По умолчанию метасимволы в именах пакетов раскрываются, что приводит к удалению всех пакетов, имена которых соответствуют шаблону. Опция `-G` за прещает интерпретацию метасимволов Это позволяет удалять пакеты с не стандартными именами
- `-i` — заставляет команду запрашивать подтверждение перед удалением каждого пакста
- `r` — включает режим рекурсивного удаления, при котором удаляется вся це почка зависимостей Используйте эту опцию с осторожностью
- `x` — заставляет команду интерпретировать имя пакета как регулярное выра жение. Это позволяет удалять все пакеты, имена которых соответствуют шаб лону отбора

Имеется также опция для безусловного удаления пакстов `a` Она чрезвычайно опасна по этому лучше избегать ее

Обычно команда вызывается без дополнительных опций. Если нужно удалить группу пакетов (особенно при использовании регулярных выражений или опции `-r`) рекомендуется указывать опцию `i`, чтобы команда запрашивала подтверждение по каждому пакету Это выглядит примерно так

```
# pkg_delete -ir wmcdplay 1 0b1
delete wmcdplay-1.0b1? y
```

В этом примере удаляется пакет `wmcdplay 1 0b1` и все зависящие от него про граммы (таковых не оказалось) Поскольку установлена опция `-i`, команда `pkg delete` запрашивает подтверждение перед удалением пакета. Если удаляемых пакетов несколько, для каждого из них требуется отдельное подтверждение

Команда `pkg_delete` может удалить пакет, инсталлированный через коллекцию переносимых приложений, но лучше перейти в соответствующий каталог коллекции и ввести **`make deinstall`** Эта команда делает примерно то же самое, но учитывает различия между двумя способами инсталляции программ

Если программа инсталлировалась по другому, например компилировалась вруч ную, процесс ее удаления будет отличаться Чаще всего приходится самостоятельно искать инсталлированные файлы и удалять их самостоятельно. Это довольно утоми тельный процесс нужно найти с помощью команды `find` файлы, имена которых как то связаны с программой, и выполнить команду `pkg_info`, чтобы определить их при надлежность Если у программы есть инсталляционный сценарий, поищите в нем информацию о том, как называются файлы и где они находятся. Некоторые коммер ческие программы поставляются с инсталляторами, которые умеют выполнять и де инсталляцию, либо устанавливаются в виде пакетов FreeBSD Их проще деинсталли ровать

Обновление программы

Иногда требуется заменить существующую программу более новой версией. В слу чае пакетов соответствующая команда называется `pkg_update` Ее синтаксис таков:

```
pkg_update [-nv] [-r старый_пакет] новый_пакет
```

Опции `n` и `v` имеют то же назначение, что и в команде `pkg_add`. Опция `r` *старый_пакет* обычно не требуется, поскольку команда сама определяет, какой пакет требует замены. Только если было инсталлировано несколько версий программы в разные каталоги, понадобится указать какую из них следует обновить. Следующая команда обновляет пакет Samba

```
# pkg_update samba-2.2.2.1.tgz
```

Более мощная утилита, умеющая работать с переносимыми приложениями, называется `portupgrade`. Она инсталлируется либо как пакет (категория `sysutils`), либо как переносимое приложение (каталог `/usr/ports/sysutils/portupgrade`). Во втором случае программа зависит от нескольких дополнительных пакетов, которые могут отсутствовать в системе, поэтому пакетную версию быстрее инсталлировать. В общем случае утилита вызывается так

```
# portupgrade -P samba
```

Эта команда обновляет пакет Samba. Опция `-P` заставляет утилиту использовать пакетный формат. Если соответствующий пакет не найден или опция `-P` не указана, будет использована коллекция переносимых приложений. Ниже перечислены другие интересные опции:

- `-a` — заставляет систему обновить все инсталлированные пакеты. Это может занять очень много времени, особенно в отсутствие опции `-P`.
- `-f` — включает режим принудительного обновления, в котором можно устанавливать более старые версии программ.
- `-i` — По умолчанию утилита `portupgrade` не спрашивает разрешения перед инсталляцией нового пакета. Опция `-i` отменяет такое поведение.
- `-n` — заставляет утилиту сообщить о планируемых действиях, но не выполнять само обновление.
- `-N` — позволяет инсталлировать пакет, отсутствующий в системе.

Достоинством утилиты `portupgrade` является то, что она отслеживает зависимости, обновляя их и внося исправленную информацию в пакетную базу данных. Есть еще утилита `pkgdb`. Если ввести `pkgdb -F`, FreeBSD проверит базу данных и устранит найденные несоответствия, в случае необходимости запросив нужную информацию у пользователя.

Проверка аутентичности программ

На сегодняшний день широкое распространение зловредного кода превратилось в серьезную проблему. Имеются в виду программы, выполняющие разного рода несанкционированные операции, такие как слежение за действиями пользователя или стирание жесткого диска. Это могут быть вирусы, "черви" и "троянские кони". В Windows подобные программы часто распространяются по электронной почте. Во FreeBSD такое тоже возможно, но механизм учетных записей служит надежной защитой системы. Более опасны поддельные программные пакеты. Хакер может добавить несанкционированный код в популярную программу и обманным путем поместить ее в хранилище, где она станет доступна для загрузки и инсталляции. Иногда злоумышленник намеренно заявляет о том, что им разработана полезная программа, хотя в действительности предназначена для других целей ("троянский конь").

Существует ряд способов защиты от зловредных программ.

- **Работайте только с надежными узлами.** Можно существенно снизить риск появления "зараженных" программ, если загружать программы только с ведущих узлов, таких как основной узел FreeBSD и его официальные "зеркала". Менее надежным источником являются неофициальные "зеркала", остальные узлы еще более ненадежны. Но даже такая мера не дает 100-процентную гарантию. Хакер может взломать узел и подменить его файлы либо исказить записи сервера имен, чтобы направить пользователей на другой узел.
- **Инсталлируйте только проверенные пакеты.** Старайтесь работать только с широко известными программами, в частности с теми, которые входят в состав FreeBSD. За годы эксплуатации они были многократно проверены и приобрели хорошую репутацию. Относительно неизвестная программа может оказаться "троянским конем".
- **Проверяйте контрольные суммы MD5.** Разработчики многих программ снабжают их контрольными суммами MD5. Это 32-байтовые шестнадцатеричные числа, полученные путем хеширования содержимого файла. Теоретически можно создать модифицированный файл с такой же контрольной суммой MD5, но на практике это нереально. Если загруженная программа имеет хеш-код, заявленный автором, можно быть уверенным в том, что она не подделана.

В любом случае невозможно на 100 процентов гарантировать безопасность инсталлированной программы, поэтому бдительность не бывает лишней. Программа `sysinstall` загружает код только из надежных источников (инсталляционный компакт-диск или официальные "зеркала" FreeBSD) и инсталлирует только пакеты, являющиеся официальными компонентами FreeBSD. Исходные коды переносимых приложений тоже загружаются с официальных Web- или FTP-узлов. На локальном диске для них хранятся контрольные суммы MD5, которые обязательно проверяются. Наиболее рискованный метод инсталляции — использование команды `pkg_add`, особенно если пакет получен с неофициального узла без контрольной суммы MD5. Для проверки контрольных сумм предназначена команда `md5`:

```
% md5 zile-1 6.1.tgz
MD5 (zile-1.6.1 tgz) = c98c146326723f8526ca6701b4bf540
```

Сравните полученную 32 символьную строку со значением, которое указано на официальном Web-узле программы или в другом заслуживающем доверия источнике. У разных версий программы контрольные суммы, разумеется, будут отличаться.

Резюме

Инсталляция программного обеспечения — одна из важнейших обязанностей системного администратора FreeBSD. Существуют различные способы инсталляции и обслуживания программ. Как правило, они основаны на использовании базы данных с информацией о пакетах, имеющихся в системе. Можно также компилировать программу вручную, если доступен ее исходный код, или, как в случае коммерческих приложений, пользоваться специализированными инсталляторами. Кроме того, в системе есть ряд утилит, предназначенных для управления пакетами, в частности для их удаления, обновления, проверки наличия и т.д.

Полный справочник по

Глава 12

Конфигурирование ядра

Методы, описанные в главе 11, позволяют обновлять большинство программ FreeBSD. Но одна программа заслуживает особого внимания: *ядро*. Именно ядро лежит в основе операционной системы. Оно является центральной управляющей программой, предоставляя остальным программам доступ к аппаратным ресурсам компьютера, разрешая конфликты между программами и т.д. Это означает, что ядро нужно конфигурировать, обновлять и запускать отдельно от остальных программ. Ему и посвящена настоящая глава.

Глава начинается с общих сведений о ядре, чтобы читатели лучше уяснили, какую роль играет ядро системы. Далее рассматриваются способы модификации ядра, в том числе принципы его компиляции. Это важно знать системным администраторам. В конце главы рассказывается о том, как обновить версию ядра.

Общие сведения о ядре

Прежде чем осваивать процедуры конфигурирования и инсталляции ядра, необходимо понять, что оно делает. Роль ядра настолько велика, что при его обновлении часто приходится перезагружать компьютер. В случае других программ это делается редко.

Роль ядра во FreeBSD

Ядро — самая низкоуровневая программа компьютера. Для большинства устройств это единственная программа, которой разрешен прямой доступ к ним. Такой доступ осуществляется с помощью *драйверов*, написанных для конкретных аппаратных компонентов, например Ethernet-платы VIA Rhine или последовательного интерфейса 16550 RS-232. Программы обращаются к ядру за получением доступа к сетевой плате или последовательному порту, а ядро транслирует эти вызовы в обращения к конкретным устройствам, подключенным к компьютеру.

На заметку

В ходе начальной загрузки компьютера, а в некоторых операционных системах и после этого, выполняется более низкоуровневый код, чем ядро. Это BIOS (Basic Input/Output System — базовая система ввода-вывода), код которой служит интерфейсом между операционной системой и аппаратной частью компьютера. На компьютерах семейства *x86* код BIOS довольно примитивен, поэтому в большинстве современных систем, включая FreeBSD, он используется лишь на ранних этапах начальной загрузки, так как остальную работу гораздо лучше выполняет ядро.

Ядро не только формирует интерфейс доступа к аппаратным устройствам, но и выполняет другие важные функции. В частности, оно реализует файловые системы, чтобы программы могли осуществлять чтение и запись файлов, не заботясь о том, в каких секторах диска это происходит. Ядро управляет памятью, чтобы процессы не могли получать доступ к данным друг друга, и контролирует доступ к сети, чтобы программам не нужно было самостоятельно создавать пакеты данных, передаваемых через сетевой интерфейс.

Что касается доступа к аппаратным устройствам, то здесь есть исключения. Некоторым программам, например X-серверу (описывается в главе 13), разрешается более свободный доступ к графической подсистеме. Другие устройства, такие как модемы, принтеры и сканеры, тоже контролируются программами пользовательского уровня (т.е. программами, которые не являются частью ядра). Но эти программы должны взаимодействовать со своими устройствами через порты, которые, в свою очередь, находятся под управлением ядра.

Инствлляция и модификация ядра

Ядро создается при инсталляции системы. Оно хранится на диске в файле `/boot/kernel/kernel` (до появления FreeBSD 5.0 это был файл `/kernel`) и вызывается системным загрузчиком в момент запуска системы, как описывалось в главе 6. Чтобы инсталлировать новое ядро, нужно записать в каталог ядра новый файл и передать загрузчику ссылку на него. Можно просто заменить существующий файл ядра. В любом случае ядро приходится предварительно компилировать, как описывается в разделе "Перекомпиляция ядра". Иногда, чтобы изменить функции ядра, достаточно загрузить новый модуль.

Не удаляйте старое ядро, пока не убедитесь в том, что новое работает без ошибок. FreeBSD позволяет выбрать нужное ядро на этапе начальной загрузки, так что можете предварительно протестировать новое ядро, указав его имя в строке приглашения загрузчика.

Когда требуется модификация ядра? Есть несколько причин:

- **Новые драйверы.** Не все драйверы включены в стандартное ядро. Это могут быть как аппаратные драйверы (например, для нового контроллера EIDE), так и драйверы новых сетевых протоколов, файловых систем и других высокоуровневых подсистем.
- **Обновления и исправления ошибок.** Ни одна программа не лишена ошибок, и FreeBSD — не исключение. В ядре системы время от времени выявляют ошибки, которые приходится устранять. Обновления могут также способствовать повышению производительности системы.
- **Уменьшение объема занимаемой памяти.** Стандартное ядро FreeBSD включает драйверы множества компонентов, которые могут не использоваться системой. Такие драйверы загружаются в память, но не выполняют никакой полезной работы. Можно повысить производительность системы, сократив размер ядра за счет удаления ненужных компонентов. Это особенно актуально в старых системах с ограниченным объемом оперативной памяти. В современных системах с объемом ОЗУ 128 Мбайт и выше эффект от сокращения размера ядра на мегабайт вряд ли будет ощутим.
- **Добавление новых функциональных возможностей и оптимизация работы.** Стандартное ядро поддерживает большинство аппаратных компонентов, но некоторые параметры могут оказаться неоптимальными. Например, по умолчанию поддерживается только один процессор, поэтому в многопроцессорной системе ядро нужно перекомпилировать.
- **Аппаратные конфликты.** Иногда случаются и аппаратные конфликты. Они не дают системе работать корректно, поскольку драйвер одного из устройств мешает работе другого устройства. Решением проблемы является удаление некорректного драйвера.

Во всех вышеперечисленных случаях требуется перекомпилировать ядро, хотя некоторые аппаратные конфликты можно устранить, модифицируя параметры загрузки ядра. Иногда драйверы реализуются в виде *загружаемых модулей*, которые представляют собой компоненты ядра, хранящиеся отдельно от него. Как правило, эти модули можно загружать без перезапуска системы, что очень удобно.

Параметры загрузки ядра

Проще всего модифицировать работу ядра, передав ему необходимые параметры в процессе загрузки Подробнее об этом рассказывалось на с 112 Напомним, что в строке приглашения загрузчика нужно нажать любую клавишу, кроме <Enter> Две команды, понимаемые загрузчиком, особенно важны

- `boot [-опции] [ядро]` позволяет загрузить указанное ядро и передать ему необходимые параметры
- `load [-t тип] модуль` — загружает указанный модуль ядра Она нужна только в том случае, когда нестандартное устройство, например SCSI-контроллер обеспечивает загрузку системы Для других периферийных устройств используются команды, описанные в следующем разделе

С помощью команды `boot` можно загрузить новое ядро вместо стандартного (`/boot/kernel/kernel`) или восстановить старое ядро, если новое работает с ошибками На процесс загрузки ядра влияют опции команды Например, опция `-s` задает загрузку в однопользовательском режиме, который применяется для аварийного восстановления системы Опция `-a` заставляет систему спросить, какое устройство должно монтироваться как корневая файловая система Это удобно, если схема разбивки на разделы поменялась

Загрузка модулей ядра

Если необходимо включить в ядро функциональные возможности которые изначально отсутствовали, то наиболее простым решением будет загрузка соответствующего модуля ядра. Без этого не обойтись, когда система не обнаруживает имеющееся аппаратное устройство.

Чтобы определить статус загруженных модулей воспользуйтесь командой `kldstat`

```
$ kldstat
Id Refs Address    Size     Name
 1   7 0xc0100000 3ef2dc   kernel
 2   1 0xc1d0d000 16000    linux.ko
```

Обычно команда сообщает лишь о нескольких модулях Это может быть модуль `kernel`, под которым скрывается собственно файл ядра (в приведенном примере присутствует также модуль `linux.ko`, реализующий поддержку Linux программ). Тот факт, что остальные модули не отображаются, не является причиной для беспокойства. Если они корректно встроены в ядро, то команда `kldstat` по умолчанию не сообщает о них Опция `-v` заставляет команду сообщать о компонентах каждого модуля Например, у модуля `kernel` могут быть десятки и даже сотни компонентов Если нужно получить информацию о конкретном модуле, воспользуйтесь опциями `-i` *идентификатор* и `-n` *имя*. Они задают идентификатор модуля (отображается в первой колонке) и его имя (последняя колонка) Вот как могут быть отображены сведения только о модуле `linux.ko`

```
$ kldstat -i 2
$ kldstat -n linux.ko
```

Чтобы загрузить нужный модуль, вызовите команду `kldload` указав имя модуля Например, следующая команда загружает модуль `snd_gusc.ko`

```
# kldload snd_gusc.ko
```

Этот модуль реализует поддержку звуковых плат Gravis UltraSound. В данном случае назначение модуля можно примерно понять по его имени, но иногда названия модулей ничего не говорят, поэтому приходится обращаться к документации. Модули обычно находятся в каталоге `/boot/kernel` или `/boot/modules`. В остальных случаях придется указывать полный путь к файлу. При наличии опции `v` команда `kldload` сообщит о выполненных действиях и о том, какой идентификатор присвоен модулю.

Для выгрузки модулей предназначена команда `kldunload`. В качестве аргумента можно указать имя модуля или его номер. Во втором случае требуется опция `-i`:

```
# kldunload -i 7
```

Преимуществом модулей является то, что они позволяют не перекомпилировать ядро и не перезагружать компьютер. Впрочем, ядро поддерживает только те модули, которые присутствовали на момент его компиляции. Если добавляется новый модуль, ядро необходимо перекомпилировать.

Перекомпиляция ядра

Самая сложная задача, связанная с конфигурированием ядра, — это его компиляция. В процессе компиляции можно настроить многочисленные параметры ядра, удалить ненужные драйверы и включить функциональные возможности, которые отсутствуют в стандартном ядре FreeBSD. Системный администратор должен уметь компилировать ядро, так как это позволяет оптимизировать конфигурацию системы, а иногда это единственный способ задействовать необходимые функциональные возможности.

Получение исходного кода ядра

Прежде чем приступать к компиляции ядра, нужно найти его исходный код. В первую очередь загляните в каталог `/usr/src/sys`. Если он существует, последующие действия не требуются. В противном случае воспользуйтесь программой `sysinstall`, которая загрузит исходный код ядра с инсталляционного компакт-диска или из Internet. Последовательность действий должна быть такой:

1. Перейдите в режиме суперпользователя и введите **`/stand/sysinstall`**, чтобы запустить программу `sysinstall`.
2. Выберите пункт `Configure`, чтобы перейти в меню конфигурирования FreeBSD (`FreeBSD Configuration Menu`).
3. Выберите пункт `Distributions`. В результате появится меню, в котором можно выбрать название инсталлируемого дистрибутива. Дистрибутив — это набор программ или других файлов, например исходных кодов системы.
4. Выберите пункт `src`. Появится меню со списком содержимого дистрибутива.
5. Выберите `sys`. Это пакет исходных кодов ядра, как указано в комментарии справа.
6. Дважды выберите `OK`, чтобы перейти в меню выбора инсталляционного источника.
7. Задайте источник, из которого будет инсталлирован исходный код. В зависимости от сделанного выбора могут понадобиться дополнительные установки, например адрес FTP-узла. Далее программа загрузит исходный код и запишет его в каталог `/usr/src/sys`, после чего снова появится меню `FreeBSD Configuration Menu`.
8. Выберите `Cancel` а затем `Exit Install`, чтобы выйти из программы `sysinstall`.

В каталоге /usr/src/sys будет содержаться один файл (Makefile) и несколько подкаталогов для конкретных типов драйверов и средств поддержки, например fs (файловые системы) и pci (платы PCI). Все это необходимо для разработчиков ядра. Нас же интересует лишь конфигурационный файл.

Настройка ядра

В процессе перекомпиляции не происходит автоматическая настройка ядра на оптимальную производительность. Соответствующие установки делаются в конфигурационном файле. Конфигурационный файл стандартного ядра называется /usr/src/sys/i386/GENERIC (компонент i386 будет другим, если компьютер не относится к платформе x86). В нем устанавливается большинство параметров ядра, вследствие чего оно получается очень громоздким, примерно 4 Мбайт во FreeBSD 5.0.

Конфигурационный файл GENERIC можно редактировать напрямую, но лучше создать его копию, например MYGENERIC (имена конфигурационных файлов ядра традиционно записываются прописными буквами, хотя это не обязательно). Модифицированный файл можно поместить в другой каталог, скажем /root, а затем создать на него символическую ссылку в каталоге /usr/src/sys/i386. Это позволит удалять и переинсталлировать пакет исходных кодов ядра, не затрагивая конфигурационный файл.

Конфигурационный файл начинается с серии комментариев — строк, помеченных символом '#'. Далее расположена такая группа строк.

```
machine     i386
cpu         I486_CPU
cpu         I586_CPU
cpu         I686_CPU
ident       GENERIC
maxusers    0
```

Строка machine идентифицирует процессорную платформу — i386 для компьютеров семейства x86. Ее менять не нужно. В следующих строках определяются три возможных семейства: 486, 586 (Pentium) и 686 (Pentium II и выше). Превратите в комментарий те строки, которые не соответствуют действительности. Благодаря этому повысится эффективность ядра. В строке ident идентифицируется само ядро. Здесь можно задать название новой конфигурации. Строка maxusers используется для настройки определенных параметров системы, например предельно допустимого числа процессов (устанавливается равным 20+16×maxusers). Если значение maxusers равно 0, FreeBSD версий 4.5 и выше осуществит автоматическую настройку.

На заметку

Несмотря на название, значение maxusers не ограничивает число пользователей, которым разрешено регистрироваться в системе. Оно определяет другие параметры, косвенно влияющие на количество пользователей системы.

До появления FreeBSD 5.0 в конфигурационный файл ядра включались различные "подсказки", касающиеся конфигурации аппаратных устройств. Это выглядело следующим образом:

```
device      ata0  at isa? port IO_WD1 irq 14
```

Во FreeBSD 5.0 конфигурационный файл упрощен, а все "подсказки" перемещены в отдельный файл /boot/device.hints. Теперь они выглядят так.

```
hint.ata.0 at="isa"
hint.ata.0 port="0x1F0"
hint.ata.0 irq="14"
```

В соответствующей строке конфигурационного файла эта информация отсутствует.

```
device      ata
```

Если необходимо вернуться к старому стилю описания, удалите символ комментария из следующей строки:

```
#hints      "GENERIC.hints"
```

Сами "подсказки" тоже можно менять. Стандартные "подсказки" хорошо работают в большинстве систем, и корректировать их приходится лишь в случае очень старых плат ISA с нестандартными установками, такими как адрес порта ввода-вывода и номер прерывания.

Остальные строки конфигурационного файла начинаются со слова `options` или `device`. Они обозначают компоненты ядра, не связанные (`options`) или связанные (`device`) с аппаратными устройствами. Иногда такое разделение оказывается довольно нечетким. Например, компонент, реализующий поддержку симметричной многопроцессорной обработки, обозначен как `options`. Большинство строк заканчивается комментарием, в котором поясняется их назначение. Рассмотрим примеры:

```
options     INET          #InterNETworking
options     FFS           #Berkeley Fast Filesystem
device      ata
device      atadisk       # ATA disk drives
device      atapicd       # ATAPI CDROM drives
```

В этих строках включается поддержка протоколов TCP/IP, стандартной файловой системы FFS (Fast File System), дискового интерфейса ATA (он же EIDE), драйверов ATA-дисков и драйверов ATA-дисководов CD-ROM. Внимательно просмотрите файл и поставьте символ комментария в начале тех строк, которые не нужны. Но не слишком увлекайтесь, поскольку некоторые компоненты лишь кажутся несущественными, а на самом деле это далеко не так. Придерживайтесь правила: если назначение компонента непонятно, лучше его не трогать.

Некоторые строки по умолчанию являются комментарием. Иногда это связано с тем, что соответствующий компонент реализует дублирующийся или устаревший драйвер устройства. Иногда драйвер конфликтует с другими устройствами или вызывает снижение производительности системы. Чаще всего отключаются компоненты SCSI и RAID, поскольку соответствующие адаптеры встречаются редко. Большинство Ethernet-устройств тоже отключается, так как обычно имеется всего одна сетевая плата.

Подробнее формат конфигурационного файла описан в файле `NOTES` (до появления FreeBSD 5.0 он назывался `LINT`). В нем даны расширенные комментарии к большинству конфигурационных параметров, а также приведены сведения о дополнительных параметрах, предназначенных для конфигурирования экзотических компонентов и драйверов. К примеру, компонент, реализующий поддержку файловой системы ext2fs (см. главу 4), упомянут в файле `NOTES`, но в стандартном файле `GENERIC` он отсутствует даже в виде комментария.

Компиляция

Для компиляции ядра нужно выполнить три команды:

```
# cd /usr/src
# make buildkernel KERNCONF=конфигурационный_файл
# make installkernel KERNCONF=конфигурационный_файл
```

В качестве значения аргумента `KERNCONF` задается имя конфигурационного файла ядра. Вторая команда выполняется от нескольких минут до нескольких часов, в зависимости от скорости процессора и количества включенных компонентов. По мере компиляции большого количества файлов ядра и объединения их в единое целое команда будет отображать информацию о них. Третья команда инсталлирует новое ядро, переименовывая прежний файл `/boot/kernel/kernel` в `kernel.old`. Таким образом, после перезагрузки будет подключено новое ядро. Старое ядро тоже останется доступным, и его можно будет подключить с помощью команды `boot` загрузчика.

Совет В целях повышения безопасности скопируйте исходное ядро в другой каталог, например `/root`, прежде чем выполнять компиляцию. Это позволит иметь резервный вариант на случай многократной перекомпиляции ядра.

Компиляция ядра нередко завершается неудачей, особенно если это делает новичок. Зачастую ошибка связана с отсутствием необходимых средств разработки. Просмотрите сообщение об ошибке и инсталлируйте с помощью программы `sysinstall` недостающий компонент. Не забывайте также, что для компиляции ядра необходимо иметь привилегии суперпользователя.

Подключение нового ядра

После перезагрузки компьютера новое ядро будет использовано по умолчанию. На всякий случай проверьте сообщения, выдаваемые командой `dmesg`. Среди них должны присутствовать следующие строки:

```
FreeBSD 5.0-20011211-CURRENT #0: Tue Dec 11 10:29:22 GMT 200
    root@usw2.freebsd.org:/usr/src/sys/i386/compile/GENERIC
```

Здесь указаны версия FreeBSD (`5.0-20011211-CURRENT` – это версия для разработчиков, предшествующая официальному выпуску FreeBSD 5.0), дата и время компиляции ядра (11 декабря 2001 г.), пользователь, осуществивший компиляцию (`root@usw2.freebsd.org`), и конфигурационный сценарий (`GENERIC`). Если эти значения соответствуют предыдущей версии ядра, то, очевидно, что-то произошло не так, как нужно: либо команда `make installkernel` завершилась неудачно, либо ядро не удалось скомпилировать.

До появления FreeBSD 5.0 иногда приходилось выполнять дополнительные действия, чтобы задействовать все функциональные возможности нового ядра. Самое главное, нужно было создавать файлы устройств для новых аппаратных драйверов. К счастью, это делается довольно просто:

```
# cd /dev
# ./MAKEDEV
```

В более поздних версиях используется специальная виртуальная файловая система `/dev`, которая создается динамически и содержит файлы устройств для всех аппаратных компонентов, поддерживаемых текущим ядром.

Обновление ядра

Одна из основных причин, по которой приходится перекомпилировать ядро, заключается в обновлении его версии. В целом это происходит так, как было описано выше, но есть ряд нюансов, которые следует учитывать.

Когда нужно обновлять ядро

Причин для обновления ядра может быть несколько:

- **Исправления ошибок**. Если в текущем ядре выявлены ошибки, то их устранение может быть продиктовано требованиями безопасности. Эти ошибки особенно серьезны, так как влияют на стабильность всей системы в целом.
- **Новые драйверы**. В новой версии ядра могут присутствовать драйверы устройств, которые не поддерживались в старой версии. Кроме того, в новых ядрах иногда улучшают существующие драйверы, например повышают их производительность.

- **Новые программные интерфейсы** В новом ядре могут быть реализованы удобные программные интерфейсы Например упоминавшаяся выше файловая система /dev — это компонент ядра

Обновление ядра предполагает и обновление его исходных кодов а это требует дополнительных усилий

На заметку Разработка FreeBSD ведется по двум направлениям, называемым ветвями ветвь STABLE и ветвь CURRENT Большинство систем относится к первой ветви, т е это полностью официальные, завершенные версии Вторая ветвь объединяет предварительные бета-версии Ядро помеченное как CURRENT, используется в основном разработчиками

Инсталляция исходных кодов

Рассмотренный ниже метод обновления ядра напоминает обновление коллекции переносимых приложений (см. главу 11)

1. Следуйте инструкциям, приведенным на с 266 по поводу инсталляции и настройки пакета CVSup, но не вводите команду `cvsup`.
2. Отредактируйте файл `/etc/cvsupfile` В главе 11 требовалось превратить в комментарий строки, начинающиеся с префикса `src-`. Но для обновления исходных кодов ядра необходимо, чтобы в файле присутствовала следующая строка.

```
src-sys
```

3. Если другие исходные коды обновлять не нужно превратите в комментарий соответствующие строки файла `/etc/cvsupfile`
4. Введите следующую команду:

```
# /usr/local/bin/cvsup -g -L 2 /etc/cvsupfile
```

Эта команда обновит каталог исходных кодов ядра Весь процесс может занять от нескольких секунд до нескольких часов, в зависимости от скорости сетевого соединения

Компиляция и подключение нового ядра

Новое ядро компилируется и подключается так же, как и обычное ядро, загружаемое с инсталляционного компакт диска Необходимо повторить действия, связанные с конфигурированием ядра (скопировать файл GENERIC и отредактировать его), поскольку новое ядро может содержать дополнительные драйверы или драйверы с другими именами Есть и более простой вариант: сравнить с помощью команды `diff` старый и новый конфигурационные файлы и скопировать в имеющийся файл строки в которых активизируются отсутствующие компоненты (команда `diff` помечает старые строки символом '<', а новые — символом '>').

Резюме

Ядро FreeBSD управляет работой всех программ, выполняющихся в системе. Вот почему так важно регулярно его обновлять и корректно конфигурировать. Перекомпиляция ядра с учетом особенностей конкретной системы может привести к повышению ее производительности и стабильности Процесс компиляции несложен: нужно скопировать конфигурационный файл ядра отредактировать его с учетом необходимых настроек и выполнить несколько команд После этого следует перезагрузить компьютер, чтобы изменения вступили в силу.

Полный справочник по

Современные пользователи предпочитают работать за компьютером в графической среде, когда приложения выполняются в окнах, а большинство операций — с помощью мыши. Такая среда есть и во FreeBSD. Она называется *X Window System*, или сокращенно *X-среда*. В главе 23 будет рассказываться о работе в этой среде и ее настольных оболочках на пользовательском уровне. Здесь же мы поговорим о низкоуровневом конфигурировании графической подсистемы.

X Window System — необычная графическая среда, в сравнении с графическими оболочками таких операционных систем, как Windows и Mac OS. Чтобы понять ее возможности и особенности, нужно изучить принципы ее функционирования. Низкоуровневое конфигурирование X-среды осуществляется путем редактирования одного файла `XF86Config`. В нем разрешается менять настройки экрана, мыши, шрифтов и многое другое. Некоторые установки, например скорость перемещения указателя мыши и скорость повтора, поддерживаемая клавиатурой, могут задаваться каждым пользователем по-своему. В конце главы будет показано, как запустить несколько X сеансов. Все они могут быть сконфигурированы по-разному, например с различными разрешением и цветовой разрядностью.

Основы X-среды

X Window System уникальна по нескольким причинам. Во-первых, она имеет очень небольшой размер в сравнении с другими популярными графическими средами. Это достигается за счет дополнительных компонентов, таких как диспетчеры окон и наборы элементов управления. Вторая причина — встроенная сетевая поддержка. Взаимодействие программ в рамках X-среды основано на сетевых протоколах, даже если все программы работают на одном компьютере. Эта особенность значительно упрощает использование X Window System в определенных сетевых конфигурациях, но в то же время несколько снижает производительность.

Графическая модель

В UNIX широко применяется модульный подход к разработке программ. Идея заключается в том, чтобы небольшие программы можно было комбинировать для решения более сложных задач. Подобная философия проектирования идет вразрез с принципами, принятыми в Windows и Mac OS, где программы просто разрастаются по мере расширения их функциональных возможностей. Естественно, бывают и исключения, но X Window System к ним не относится. Это по-настоящему модульная среда, отдельные компоненты которой в совокупности формируют полноценную графическую подсистему.

X-серверы

Основу X-среды составляет компонент, называемый *X сервером*. Его задача заключается в управлении монитором, клавиатурой, мышью и другими устройствами взаимодействия с пользователем. X сервер реализует ряд функций, которые могут использоваться программами для открытия окон, отображения текста и т.д. (Окна X-среды чрезвычайно просты. Они представляют собой всего лишь прямоугольные области экрана. Специальный компонент, называемый диспетчером окон, добавляет к ним все необходимые элементы интерфейса.) X-сервер содержит также средства обработки шрифтов и может принимать запросы по сети.

Во FreeBSD используется популярный X-сервер *XFree86* (`www.xfree86.org`), который доступен для множества платформ, включая другие BSD-системы, Linux, Mac OS X и даже OS/2 и Windows. FreeBSD 5.0 и более поздних версий содержит пакет XFree86 4.*x*. Во FreeBSD 4.*x* входил пакет версии 3.*x*. При переходе от одной версии к другой произошли изменения в структуре X-сервера. Прежде всего, в версии 3.*x* для разных семейств графических плат запускались разные серверы. Например, в случае плат S3 это был сервер `XF86_S3`, а в случае плат ATI Mach64 — сервер `XF86_Mach64`. В версии 4.*x* используется единый исполняемый файл `XFree86`. Он загружает драйверные модули, обслуживающие конкретные семейства графических плат. Это существенно упрощает конфигурирование сервера и разработку драйверов.

Поскольку графическая плата находится под контролем X-сервера, именно он (совместно с аппаратными компонентами самой платы) определяет производительность графической подсистемы, разрешение экрана и т.д. Иногда для повышения производительности достаточно обновить один лишь X-сервер или файл драйвера.

Наборы элементов управления

X-сервер управляет доступом к аппаратным ресурсам графической платы и реализует низкоуровневые API-функции для X-программ. С помощью этих функций программы открывают окна, отображают в них информацию и т.д. Программный интерфейс X-сервера чрезвычайно прост. В отличие от аналогичных интерфейсов Windows, Mac OS и других графических операционных систем, в нем не предусмотрены средства отображения меню, полос прокрутки, кнопок и других графических заготовок. Эта задача возлагается на другой компонент X-среды, называемый *набором элементов управления*.

Существует несколько подобных наборов для X Window System. В каждом из них элементы управления выглядят по-особому. Наиболее распространены оконные интерфейсы Motif (ранний коммерческий интерфейс), LessTif (клон Motif с открытым исходным кодом), Qt (популярный открытый интерфейс) и GTK (другой популярный интерфейс с открытым исходным кодом). В настоящее время во FreeBSD доминируют оконные интерфейсы Qt и GTK.

Конкретный набор элементов управления выбирается программистом, который пишет графическое приложение. От пользователей и системного администратора здесь мало что зависит. Они могут лишь отдать предпочтение тому или иному приложению. Иногда для работы программы необходимо инсталлировать библиотеки, содержащие исходный код элементов управления. Эта задача может обрабатываться автоматически программой `sysinstall` или подсистемой переносимых приложений.

Диспетчеры окон

X Window System содержит низкоуровневые процедуры работы с окнами, а наборы элементов управления позволяют программистам определять внутреннюю структуру окон. Еще одним компонентом X-среды является *диспетчер окон*. Он контролирует рамку окна, разрешая пользователям менять его размер и положение на экране. На рис. 13.1 изображено стандартное окно X-среды с указанием элементов управления, добавляемых диспетчером окон.

- **Строка заголовка**. *Строка заголовка*, или *строка перетаскивания*, обычно расположена вверху окна и содержит его имя. Если щелкнуть на строке заголовка и, удерживая кнопку мыши, двигать мышь, диспетчер окон будет перемещать окно по экрану.
- **Управляющие кнопки**. Для работы с окном используется несколько элементов управления. Это небольшие квадратные кнопки, расположенные на одном или обоих концах строки заголовка (рис. 13.1). Их функциональное назначение за

висит от диспетчера окон. В основном они позволяют закрывать сворачивать и разворачивать окно, иногда перемещать окно на другой виртуальный ра бочий стол.

- **Рамка.** Большинство диспетчеров окон создает рамки позволяющие масшта бировать окна. Переместите указатель мыши на край или к углу окна, и он поменяет форму Это подсказка' если щелкнуть на рамке и, удерживая кнопку мыши, двигать мышь, размер окна будет изменяться Некоторые диспетчеры реагируют на подобные операции только в правом нижнем углу окна

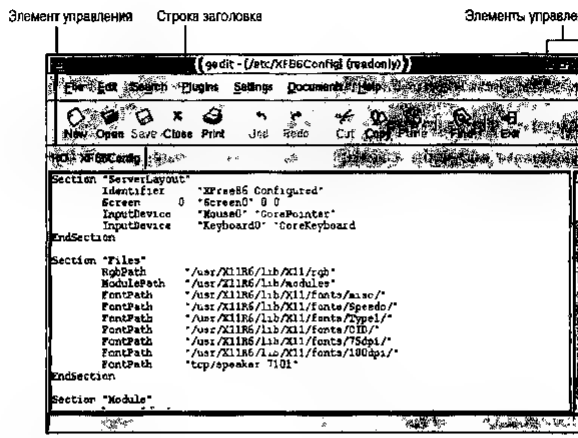


Рис. 13.1. Диспетчер окон создает декоративное обрамление окна, позво ляя изменять его положение и размер

Помимо перечисленных элементов диспетчеры окон создают дополнительные эле менты управления непосредственно на рабочем столе Если щелкнуть правой кнопкой мыши в свободной области экрана, появится меню, содержащее команды для выхода из диспетчера окон, запуска программ и др Эти функции часто перекрываются аналогичными возможностями графических оболочек (описаны в следующем подразделе)

Многие диспетчеры окон способны управлять несколькими *виртуальными рабочими столами*. Это независимые рабочие области, в каждой из которых свой набор окон Способ переключения между рабочими столами у разных диспетчеров разный. Такие же функции часто реализуются и графическими оболочками.

Важно понимать, что одни и те же окна могут выглядеть по разному в зависимости от используемого диспетчера окон Для наглядности рассмотрим рис 13.2, где изображено окно того же приложения, что и на рис 13 1. Но если первое окно (рис 13 1) создано в диспетчере IceWM, то второе (рис. 13.2) — в диспетчере WMX Последний располагает строку заголовка непривычным способом — сбоку и к тому же отображает всего одну управляющую кнопку а не четыре в отличие от IceWM В то же время внутреннее представление окна, включая строку меню, полосы прокрутки и значки панели инструментов, осталось неизменным так как оно определяется ис пользуемым набором элементов управления

Рис 13 2 Диспетчеры окон отображают окна по-разному

Не все различия можно передать посредством снимка экрана К примеру, у каждого диспетчера окон собственные правила передачи *фокуса*, т.е признака того, может ли окно реагировать на нажатия клавиш или действия мыши По умолчанию фокус обычно передается по щелчку мыши в произвольной области окна. Другая распространенная установка — активизировать окно при перемещении на него указателя мыши Как правило, активным является переднее окно, но некоторые диспетчеры могут передавать фокус окну, частично перекрытому другими окнами. Щелчок на таком окне переводит его на передний план.

Пользователь может выбрать диспетчер окон из числа тех что инсталлированы в системе О том, как это сделать, рассказывается на 296

Графические оболочки

Когда-то давно пользователи FreeBSD вызывали X Window System и диспетчер окон посредством сценариев запуска (см. главу 6) Отдельные программы запускались из терминального окна или меню диспетчера, но они были очень слабо интегрированы друг с другом, и пользователям приходилось самостоятельно добавлять ссылки на нужные утилиты, например редакторы, в программное меню (в стандартной конфигурации это мог делать системный администратор) Ситуация изменилась к лучшему с появлением графических оболочек Они представляют собой коллекции программ, использующих одинаковые наборы элементов управления и хорошо взаимодействующих друг с другом Результатом стал согласованный вид рабочего стола, конфигурирование которого осуществляется посредством специализированных утилит

На заметку

Графические оболочки создают привлекательный пользовательский интерфейс системы, но они потребляют очень много ресурсов Система, в которой работают диспетчер окон и традиционные UNIX-программы вполне может функционировать при наличии 64 Мбайт ОЗУ и даже меньше. Если же запустить оболочку KDE или GNOME, то может не хватить и 128 Мбаит

В определенном смысле графические оболочки являются преемниками диспетчеров окон поскольку запускаются вместо них. В то же время графические оболочки

можно считать расширениями диспетчеров, ведь последние всегда составляют ядро оболочки. В некоторых оболочках можно выбирать предпочтительный диспетчер.

Наиболее распространены оболочки GNOME (GNU Network Object Model Environment) и KDE (K Desktop Environment). Первый из них основан на наборе элементов управления GTK, второй реализован на основе Qt, что впрочем, не мешает запускать GNOME-приложения в KDE и наоборот.

Другие компоненты

В состав X-среды входят также следующие компоненты:

- **Утилиты конфигурирования**. Существуют различные программы, предназначенные для конфигурирования X-среды на низком уровне (задания параметров графической платы, мыши и т.д.). Некоторые из них рассмотрены на с. 294.
- **Серверы шрифтов**. Ни одна графическая среда не будет полной без набора шрифтов. X-серверы содержат встроенные средства поддержки шрифтов, настройка которых описана на с. 292. Кроме того, X Window System может работать с сетевыми серверами шрифтов, которые предоставляют шрифты всем X-станциям сети. Об этом рассказывается в главе 22.
- **Приложения**. Существует огромное множество программ, написанных для X-среды. Впрочем, многие стандартные программы FreeBSD функционируют в текстовом режиме. При работе в графическом режиме их можно запускать в терминальном окне.

Сетевая поддержка

В отличие от большинства графических сред, X Window System ориентирована на работу в сети. Можно запустить X-сервер на одном компьютере и использовать его для отображения данных, которые поступают от программы, работающей на другом компьютере, а также для передачи входных данных такой программе. Клиент-серверная терминология X Window System часто сбивает с толку, ведь мы привыкли считать, что сервер — это высокопроизводительный компьютер, скрытый от посторонних глаз, а клиент — это компьютер, за которым сидит пользователь. Попробуйте взглянуть на ситуацию с точки зрения приложения. Для программы, например текстового процессора, клавиатура, мышь и экран образуют потоки ввода-вывода, такие же, как и потоки файлового сервера, на котором хранятся данные. Следовательно, программе не важно, где именно формируется поступающий к ней входной поток. Тот факт, что пользователь сидит за сервером, для текстового процессора не принципиален.

Сетевая природа X Window System позволяет сконфигурировать производительный FreeBSD-компьютер как сервер приложений, запускаемых множеством пользователей, а гораздо менее производительные компьютеры использовать в качестве графических терминалов. Для этой цели был выпущен целый класс компьютеров, называемых *X-терминалами*. У них нет собственных вычислительных мощностей, и они служат лишь для подключения клавиатуры, мыши и монитора. Допускается также использовать одну FreeBSD-систему как X-сервер для другой такой же системы. Это дает возможность продлить срок полезной службы устаревших компьютеров. Даже системы на базе Windows и OS/2 могут функционировать в качестве X-серверов, если инсталлировать на них соответствующее программное обеспечение, например Exceed (`http://www.hcl.com/products/nc/exceed/`), Xmanager (`http://www.netsarang.com/products/xmanager.html`) или Windows-версию XFree86 (`http://sources.redhat.com/cygwin/xfree/`).

X-сервер работает медленнее, если клиент обращается к нему с другого компьютера. Но в зависимости от типа сети и приложения разница в скорости может оказаться настолько незначительной, что на нее никто не обратит внимания.

В базовой инсталляции FreeBSD конфигурируется лишь локальный X-сервер К X-приложениям такой системы нельзя получить доступ с другого компьютера В главе 21 будет рассказано о том как разрешить удаленный доступ

Конфигурирование X Window System

X-сервер устанавливается и конфигурируется в ходе инсталляции системы В некоторых случаях такая конфигурация оказывается неправильной (сервер не запускается) или неоптимальной Иногда приходится вносить изменения и в рабочую конфигурацию, например при смене мыши или графической платы В этом разделе рассказывается о том, как настраивать X-сервер Вначале будет описана общая структура конфигурационного файла `XF86Config`, а затем мы рассмотрим настройки видеоадаптера, монитора, мыши и шрифтов В завершение раздела приводится информация об утилитах, которые позволяют вносить эти изменения

На заметку Ниже описывается конфигурация пакета XFree86 4.*x* В версии 3.*x* формат конфигурационного файла был несколько иным

Предварительные замечания

Прежде чем приступать к редактированию файла `XF86Config`, полезно отключить автоматический запуск X-сервера Это даст возможность запустить сервер вручную протестировать изменения, а затем быстро завершить его работу, внести дополнительные изменения и перезапустить сервер Кстати, по умолчанию принята именно такая конфигурация Чтобы включить автоматический запуск, нужно вызвать XDM (X Display Manager) или другую аналогичную программу из файла `/etc/ttys` как описывалось в главе 6 Изначально этот файл содержит следующую строку

```
ttyv8   "/usr/X11R6/bin/xdm  nodaemon" xterm   off  secure
```

Если вместо ключевого слова `off` стоит `on`, поменяйте его и введите **`kill -HUP 1`** Эта команда переконфигурирует демон `init`, чтобы он не запускал автоматически программу XDM, но текущий экземпляр демона продолжит выполняться Если необходимо завершить работу программы XDM, выйдите из текущего сеанса и введите **`killall xdm`** в командной строке.

После внесения изменений введите **`startx`**, чтобы протестировать новую конфигурацию (попробуйте сделать это от имени непривилегированного пользователя, чтобы убедиться в доступности всех необходимых файлов) По окончании тестирования выберите в меню диспетчера окон или графической оболочки команду типа Log Out В результате X-сервер должен завершить работу, и вы окажетесь в текстовом режиме Если этого не произошло, нажмите <Ctrl+Alt+Backspace>, чтобы принудительно уничтожить X-сеанс В любом случае на экране должна появиться серия сообщений от X-сервера Среди них могут быть и такие, которые содержат важную отладочную информацию К сожалению, сообщения слишком быстро проносятся по экрану, поэтому их трудно рассмотреть Чтобы перехватить сообщения, направьте их в файл Например, вместо **`startx`** можно ввести **`startx &> startx.out`**, и сообщения окажутся в файле `startx.out`

Базовая структура файла XF86Config

Файл `XF86Config` находится в каталоге `/etc/X11` Это обычный текстовый файл который можно модифицировать с помощью любого текстового редактора, описан

ного в главе 5. Перед изменением файла желательно создать его резервную копию чтобы в случае неудачи можно было восстановить исходную конфигурацию Сущест вуют также утилиты, позволяющие вносить изменения в этот файл Они описаны на с 294

Файл XF86Config разбит на разделы, каждый из которых начинается с ключевого слова Section и заканчивается ключевым словом EndSection. Ниже показан типич ный начальный раздел

```
Section "ServerLayout"
        Identifier     "XFree86 Configured"
        Screen     0   "MyScreen" 0 0
        InputDevice    "PS2Mouse" "CorePointer"
        InputDevice    "StdKeyboard" "CoreKeyboard
EndSection
```

В разделе ServerLayout определяется структура остальной части файла. Строка Identifier задает название данной конфигурации Строка Screen идентифицирует дисплей, а строки InputDevice — два устройства ввода (мышь и клавиатура) Эти строки ссылаются на последующие разделы, в которых описаны указанные устройст ва Каждый раздел, в свою очередь, может содержать ссылки на другие разделы (как в случае раздела Screen) Таким образом, типичный файл XF86Config представляет собой иерархическую структуру (рис 13 3) Чтобы проследить иерархические связи, поищите идентификаторы устройств (такие как MyScreen и PS2Mouse в показанном примере) в строках Identifier последующих разделов Некоторые разделы располо жены как бы вне иерархии, поскольку относятся ко всему X-серверу

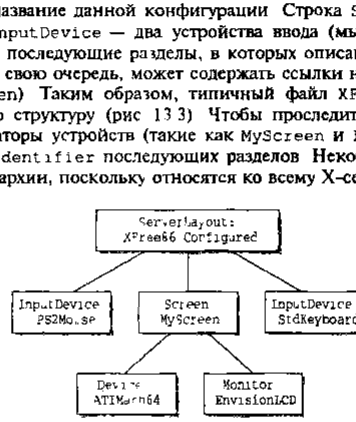


Рис 13 3 Разделы файла XF86Config форми руют иерархическое описание устройств, свя занных с X-сервером

Вот основные разделы файла

- **ServerLayout** Как только что было сказано это раздел верхнего уровня Он соединяет остальные разделы в единое целое
- **Files** Этот раздел не является частью иерархии Он определяет каталоги, в которых могут находиться важные файлы пакета XFree86, например файлы шрифтов (задаются в строках FontPath) и драйверы устройств (задаются в строках ModulePath)
- **Module**. В пакете XFree86 4 *x* применяется модульный подход к управлению драйверами, и в этом разделе задаются драйверные модули, которые должны загружаться сервером Как правило, вносить сюда изменения не требуется.
- **InputDevice** Как продемонстрировано на рис 13.3, в файле обычно присут ствуют два раздела InputDevice для клавиатуры и для мыши. Раздел клавиа

туры почти никогда не приходится менять а раздел мыши может потребовать модификации, если в процесс инсталляции системы тип мыши был задан не верно.

- **Monitor** В этом разделе содержится информация о мониторе в частности доступные диапазоны горизонтальной и вертикальной частот обновления эк рана Эта информация должна быть правильной иначе работа монитора будет нарушена

Старые мониторы можно было повредить задав слишком высокую частоту обновления. Обязательно сверяйте вводимые значения с доку ментацией к монитору

- **Device** В этом разделе задаются параметры видеоадаптера, включая драйвер платы и различные низкоуровневые установки Нередко в файле присутствует несколько таких разделов Один из них определяет оптимизированный режим для имеющейся платы, а другой — стандартный режим VGA, поддерживаемый всеми современными видеоадаптерами, но только при очень низком разреше нии Режим VGA не используется при наличии рабочей оптимизированной конфигурации.
- **Screen** В этом разделе содержатся ссылки на разделы Monitor и Device (рис 13 3) и формируется обобщенное описание дисплея Здесь же указывается стандартнвл разрядность цветовой палитры

Задание дисплея

Пакет XFree86 определяет дисплей как сочетание монитора и видеоадаптера Кон фигурация дисплея задается в разделе Screen'

```
Section "Screen"
        Identifier "MyScreen"
        Device     "ATIMach64"
        Monitor    "EnvisionLCD
        DefaultDepth 16
        SubSection "Display
              Depth      16
              Modes "1024x768   800x600   640x480"
        EndSubSection
        SubSection "Display"
              Depth      24
              Modes "800x600"  640x480"
        EndSubSection
EndSection
```

Значение параметра Identifier должно соответствовать названию дисплея, за данному в разделе ServerLayout (в данном случае MyScreen). Строки Device и Monitor ссылаются на определения видеоадаптера и монитора (приведены далее).

Раздел Screen включает один или несколько подразделов с названием Display В них задаются наборы характеристик дисплея, одна из которых — *разрядность цветовой палитры*, т.е количество битов, используемых для описания цвета пикселя Наименьшая разрядность — 1 бит Это означает, что пиксель может иметь всего два цвета (обычно черный и белый). Более распространенные значения таковы. 8 бит (256 цветов), 16 бит (65536 цветов), 24 бит (16777216 цветов) и 32 бит (4294967296 цветов). На практике используются палитры с разрядностью 16 бит и выше. При более инзкой разрядности может происходить искажение выводимых изображений.

Другим компонентом подраздела Display является спецификация поддерживае мых *графических режимов* Сами режимы определяются в другом месте (в разделе

Monitor), просто они обычно называются в соответствии с искомым разрешением, например 1024x768. Как правило, в одной строке Modes задается несколько режимов. Они проверяются по очереди пока не будет найден рабочий режим. Пользователь может выбирать остальные режимы, нажимая <Ctrl+Alt+'-'> или <Ctrl+Alt+ +'>. В XFree86 4.x часто отсутствует упоминание о графических режимах, поскольку сервер этой версии умеет автоматически задавать максимальное разрешение, поддерживаемое монитором Если нужно переопределить эту установку, укажите название режима в разделе Screen и задайте его определение в разделе Monitor

Важным элементом раздела Screen является строка DefaultDepth в которой задается стандартная разрядность цветовой палитры. Она должна соответствовать одному из значений параметра Depth в подразделах Display. Поскольку наряду с разрядностью палитры указываются и графические режимы, строка DefaultDepth определяет какие режимы будут доступны при запуске X-сервера

Настройки графической платы

Если в компьютере поменялась графическая плата или в процессе инсталляции FreeBSD плата была задана неверно, нужно модифицировать раздел Device файла XF86Config Типичный раздел выглядит так

```
Section "Device"
        Identifier "ATIMach64"
        VendorName "ATI"
        BoardName  "XPert 98
        Driver     "ati"
        Option     "DPMS"
        Option     "OffTime  "20
EndSection
```

Рассмотрим назначение его параметров.

- **Identifier** Этот параметр должен соответствовать названию видеоадаптера заданному в строке Device раздела Screen
- **VendorName и BoardName** Эти параметры устанавливают названия фирмы поставщика и самой платы. Их можно задавать произвольным образом, чтобы упростить идентификацию устройства.
- **Driver** Этот параметр идентифицирует используемый драйвер Соответствующие драйверы хранятся в каталоге /usr/X11R6/lib/modules/drivers. Имя файла драйвера определяется путем добавления строки drv.o к значению параметра Driver В рассматриваемом примере загружается драйвер ati_drv.o
- **Option** В разделе Device может присутствовать несколько строк Option которые модифицируют работу драйвера. В рассматриваемом примере опция DPMS включает *систему управления питанием монитора*, которая позволяет переводить монитор в режим с пониженным энергопотреблением при отсутствии активности пользователя. Опция OffTime заставляет систему отключать монитор после 20 минут неактивного состояния. Аналогичные установки можно делать с помощью программы xset и специальных утилит графической оболочки

При подключении к системе новой графической платы создайте для нее отдельный раздел Device и поменяйте соответствующую ссылку в разделе Screen. Это позволит быстро восстановить прежнюю конфигурацию системы, если новая плата не заработает Если вы не уверены, какой именно драйвер подходит для имеющейся платы, посетите Web-страницу http://www.xfree86.org/current/Status.html. На ней приведена информация, касающаяся поддержки графических плат пакетом XFree86.

Настройки монитора

В разделе `Monitor` определяются свойства монитора. Типичный пример выглядит так:

```
Section "Monitor"
        Identifier   "EnvisionLCD"
        VendorName   "Envision"
        ModelName    "EN-5100e"
        HorizSync    28.0 - 61.0
        VertRefresh  70    75.0
EndSection
```

Как и в других разделах, строка `Identifier` уникальным образом идентифицирует раздел. Строки `VendorName` и `ModelName` задают название фирмы-поставщика и модель монитора.

X-сервер различает две важнейшие характеристики монитора: *горизонтальную частоту обновления* и *вертикальную частоту обновления*. Они задаются с помощью параметров `HorizSync` и `VertRefresh`. Современные мониторы поддерживают целые диапазоны частот, как видно из примера. Горизонтальная частота обновления задается в килогерцах и указывает на то, как быстро монитор может отобразить горизонтальную строку пикселей. Вертикальная частота обновления задается в герцах и определяет скорость отображения полноэкранной картинки. Как правило, чем выше это значение, тем лучше, поскольку глаза меньше устают при работе за монитором. Конкретные значения можно узнать в документации к монитору.

Внимание! При наличии старого монитора вводите максимально точные значения. Старые мониторы можно повредить, если задать слишком высокую частоту обновления (новые мониторы игнорируют недопустимые режимы).

Пакет XFree86 4.x поддерживает технологию DDC (Data Display Channel — канал отображения данных), которая позволяет компьютеру запрашивать у монитора информацию о поддерживаемых режимах и параметрах конфигурации. В случае слишком старого или нестандартного монитора может понадобиться добавить в раздел `Monitor` следующие строки.

```
Mode "1024x768i"
    DotClock       45
    HTimings       1024 1048 1208 1264
    VTimings       768 776 784 817
    Flags          "Interlace"
EndMode
```

Здесь задаются параметры синхронизации для конкретного разрешения экрана — в данном случае это 1024×768 с чересстрочной разверткой (`i` в `768i` для отображения полноэкранной картинки требуется два прохода луча развертки). Лучше не экспериментировать с созданием режимов монитора, а скопировать их определения из надежного источника, например файла `XF86Config` пакета XFree86 3.x, где такие определения встречались довольно часто. Не беспокойтесь по поводу режима с чересчур высоким разрешением. X-сервер проверяет параметры `HorizSync` и `VertRefresh` и отказывается включать режим, который им не соответствует.

На заметку Чаще используется однострочная спецификация режима, задаваемая с помощью ключевого слова `Modeline`, а не `Mode`. В такую спецификацию включается та же самая информация, но без названий параметров (`DotClock`, `HTimings` и т.д.).

Настройки мыши

Параметры конфигурирования мыши находятся в разделе `InputDevice` который выглядит примерно так:

```
Section "InputDevice"
        Identifier   "PS2Mouse"
        Driver       "mouse"
        Option       "Protocol" "auto"
        Option        Device" "/dev/sysmouse"
EndSection
```

На заметку Аналогичный раздел существует и для клавиатуры однако параметры клавиатур лучше стандартизированы поэтому потребность в их модификации почти не возникает

Значение параметра `Identifier` должно соответствовать названию мыши в разделе `ServerLayout` Параметр `Driver` обычно равен `mouse`, даже если используется шаровой манипулятор, сенсорная панель или другое указательное устройство. Основные характеристики устройства задаются с помощью параметров `Option`.

- **`Protocol`** *Протокол* мыши — это набор правил, в соответствии с которыми мышь сообщает системе свои координаты В пакете XFree86 4.*x* поддерживается режим `auto`, т.е. сервер самостоятельно пытается определить протокол мыши. Если мышь не работает в этом режиме, попробуйте задать протокол вручную Наиболее распространенные значения — `PS/2`, `Microsoft` и `Logitech`. Устройства, подключаемые к порту USB или PS/2, работают по протоколу `PS/2` Остальные протоколы соответствуют устройствам которые подключаются к последовательному порту RS-232
- **`Device`.** Эта опция задает файл устройства, используемый сервером для доступа к порту мыши В рассматриваемом примере указано фиктивное устройство `/dev/sysmouse` Оно создается демоном `moused` который реализует поддержку мыши в текстовом режиме (его инсталляция описывалась в главе 2) Если демон `moused` не запущен, задайте корректный файл устройства, например `/dev/ums0` для USB-мыши, `/dev/psm0` для мыши PS/2 или один из файлов `/dev/cuaa0`–`/dev/cuaa3` для мыши RS-232 (последовательный порт)

Если неправильно выбрать протокол или устройство, X-сервер может не запуститься, не отобразить указатель мыши или начать некорректно перемещать его Нажмите <Ctrl+Alt+Backspace> чтобы завершить работу X-сервера и переконфигурируйте систему.

Совет В случае сомнении примените к файлу устройства команду `cat`, например **`cat /dev/ums0`**, и подвигайте мышь. Если на экране ничего не появляется, значит, это не тот файл Нажмите <Ctrl+C> и проверьте другой файл Описанная процедура может привести к переводу терминала в некорректный режим Чтобы восстановить установки терминала, введите **`reset`**

Добавление и удаление шрифтоа

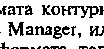
В разделе `Files` задаются каталоги в которых X сервер ищет шрифты Вот типичный пример

```
Section "Files"
        RgbPath      /usr/X11R6/lib/X11/rgb
```

```
        ModulePath    "/usr/X11R6/lib/modules"
        FontPath      "/usr/X11R6/lib/X11/fonts/misc/"
        FontPath      "/usr/X11R6/lib/X11/fonts/Speedo/"
        FontPath      "/usr/X11R6/lib/X11/fonts/Type1/"
        FontPath      "/usr/X11R6/lib/X11/fonts/75dpi/"
        FontPath      "/usr/X11R6/lib/X11/fonts/100dpi/"
        FontPath      "tcp/speaker:7101"
EndSection
```

Первые две строки можно проигнорировать. Нас интересуют только строки `FontPath`. В каждой из них задается каталог для шрифтов определенного типа. В основном это подкаталоги каталога `/usr/X11R6/lib/X11/fonts` — стандартного места хранения шрифтов во FreeBSD. Последняя строка `FontPath` не совсем обычна. В ней указан не каталог, а *сервер шрифтов*, т. е. сетевой сервер, возвращающий информацию о шрифтах по запросу X-сервера. Конфигурация сервера шрифтов описана в главе 22.

Сервер XFree86 понимает несколько типов шрифтов. Многие встроенные шрифты (предполагается, что они всегда существуют) представлены в *растровом формате*. В данном контексте это означает, что символы шрифта имеют вид матрицы из белых и черных пикселей. Растровые шрифты невелики по размеру и очень быстро отображаются, но это происходит корректно только при определенном кегле и разрешении. Чтобы увеличить размер шрифта, нужно создать новый растровый шрифт. Другой тип — это *масштабируемые* (или *контурные*) шрифты. Такой шрифт хранится в виде математического описания контуров символов. Существуют два наиболее популярных формата контурных шрифтов: PostScript Type 1 компании Adobe (известен как Adobe Type Manager, или ATM) и TrueType компании Apple. Пакет XFree86 4.*x* поддерживает оба формата, тогда как в предыдущих версиях шрифты TrueType не поддерживались.



Коллекции шрифтов часто распространяются на компакт-дисках. Сервер XFree86 работает с файлами шрифтов того же формата, что и Windows, поэтому используйте Windows-версии шрифтов, а не Macintosh версии.

Если нужно добавить шрифты, выделите для них отдельный подкаталог (его можно создать в каталоге `/usr/local`). Но просто записать шрифты в этот каталог недостаточно. Необходим сводный файл, который сообщит X-серверу об имеющихся шрифтах. Этот файл называется `fonts.dir`, и его слишком утомительно генерировать вручную. К счастью, для шрифтов формата Type 1 и TrueType существуют специальные утилиты: `typelinst` и `ttmkfdir` соответственно. Обе они доступны как в виде пакетов, так и в виде переносимых приложений (для утилиты `typelinst` пакет называется `p5 typelinst`). Вот как нужно сообщить X-серверу об инсталлированных шрифтах:

1. Перейдите в каталог новых шрифтов.
2. Если каталог содержит шрифты TrueType, введите **`ttmkfdir >fonts.scale`**. В результате будет создан промежуточный файл `fonts.scale`.
3. Если каталог содержит шрифты Type 1, введите **`typelinst`**, чтобы создать промежуточный файл `fonts.scale`.
4. Введите **`mkfontdir -e /usr/X11R6/lib/X11/fonts/encodings`**. Эта команда извлекает информацию из файла `fonts.scale` и из файлов растровых шрифтов, если они присутствуют, и создает файл `fonts.dir`.
5. Добавьте в файл `/etc/XF86Config` строку `FontPath` со ссылкой на новый каталог шрифтов.
6. Если нужно немедленно подключить новые шрифты, перезапустите X-сервер или выполните команду **`xset fp rehash`**, которая заставит сервер проверить новый набор каталогов шрифтов.

Теперь новые шрифты должны быть доступны X приложениям которые позволяют выбирать шрифты Чтобы убедиться в этом воспользуйтесь утилитой xfontsel (рис. 13 4)

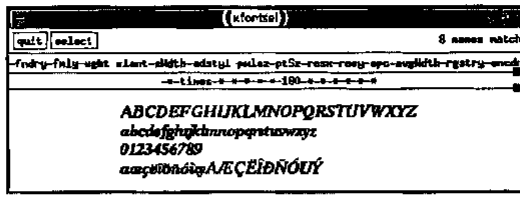


Рис 13.4. Программа xfontsel позволяет тестировать инсталлированные шрифты, задавая различные размеры и атрибуты

Совет Можно инсталлировать готовые наборы шрифтов. В коллекции переносимых приложений для этого предусмотрен раздел xll-fonts. Особенно интересен пакет urwfonts. Это набор шрифтов Type 1, выпущенных компанией URW для дублирования шрифтов PostScript принтеров

К сожалению, не все программы ориентируются исключительно на механизм инсталляции шрифтов, используемый X-сервером Просто средств обработки шрифтов X сервера недостаточно для определенных программ которые требуют, чтобы шрифты инсталлировались отдельно. Особенно это касается текстовых процессоров. В случае недоразумений читайте документацию к программе.

Чтобы деинсталлировать шрифт, нужно удалить либо его каталог, либо запись из файла fonts.dir Во втором случае поищите строку с названием шрифта Типичная запись выглядит так:

```
n0210231.pfb -Adobe-Times-medium i normal--0 0 0 0 p 0-iso8859 1
```

Некоторые шрифты имеют несколько записей для различных вариантов начертания (полужирный, курсив и т.д.), а у растровых шрифтов для каждого размера создается отдельная запись. Удалив все записи отредактируйте первую строку файла, в которой указано общее число записей.

Внимание Не удаляйте шрифты являющиеся стандартными компонентами FreeBSD. Некоторые программы предполагают наличие этих шрифтов и могут повести себя непредсказуемо в случае их отсутствия

Конфигурационные утилиты

Изменение конфигурации X-сервера путем редактирования файла XF86Config вручную — утомительная процедура, особенно для новичков Удобнее использовать специальные утилиты Ниже будут кратко описаны две из них xf86cfg и собственно сервер XFree86

Сервер XFree86 4 *x* умеет опрашивать аппаратные устройства и генерировать файл XF86Config Чтобы воспользоваться этой возможностью, введите **XFree86 -configure**. Эта команда отображает массу статусной информации и создает файл /root/XF86Config.new, который можно взять за основу для дальнейшей модификации Для проверки файла выполните команду **XFree86 -xf86config /root/**

XF86Config.new, которая запустит X-сервер, назначив ему новый конфигурационный файл Вполне вероятно, что при первом запуске сервер будет работать некорректно, поскольку функции автоматического конфигурирования часто неправильно распознают определенные устройства, например мышь, управляемую демоном moused

Если сгенерированного файла XF86Config.new недостаточно или нужно модифицировать существующий файл, попробуйте отредактировать его с помощью программы xf86cfg. Она является частью пакета XFree86 4.x и имеет графический интерфейс (рис. 13 5) Естественно, это подразумевает, что на конфигурируемом компьютере уже запущен X-сервер Таким образом, программу xf86cfg можно использовать для настройки рабочей конфигурации либо для создания новой конфигурации перед тем, как будут внесены изменения в параметры графической платы Программа вызывается следующим образом.

```
# xf86cfg  xf86config конфигурационный_файл
```

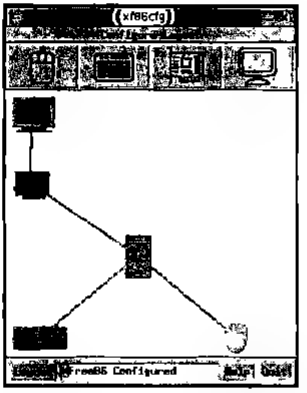


Рис 13 5 Программа xf86cfg отображает иерархию устройств ввода-вывода

Аргумент *конфигурационный_файл* должен ссылаться на файл /etc/XF86Config или на тестовый файл типа /root/XF86Config.new, созданный сервером XFree86 при наличии опции -xf86config. Чтобы сконфигурировать необходимый компонент щелкните правой кнопкой мыши на его изображении в окне программы и выберите в раскрывшемся меню пункт Configure. Появится диалоговое окно с параметрами, относящимися к выбранному устройству Разрешается также добавлять новые устройства, щелкая на их значках в верхней части окна Для подключения добавленного устройства щелкните правой кнопкой мыши на его изображении и выберите в появившемся меню команду Enable Таким способом можно подключать к рабочей станции несколько мониторов, а также подключать внешнюю USB клавиатуру или мышь к ноутбуку

Пользовательские настройки

В предыдущем разделе рассматривались глобальные конфигурационные настройки X-сервера. Они в основном задаются системным администратором. Но есть ряд параметров, которые могут устанавливаться пользователями по своему усмотрению. Наиболее важные из них касаются выбора диспетчера окон и графической оболочки. К более специфичным параметрам относятся скорость перемещения указателя мыши и скорость повтора, поддерживаемая клавиатурой. Системный администратор устанавливает стандартные значения этих параметров, но пользователи вольны переопределять их.

Выбор диспетчера окон и графической оболочки

В главе 6 описывались различные сценарии запуска, в том числе два сценария, используемые для вызова диспетчера окон. `.xinitrc` и `.xsession`. Эти сценарии находятся в начальном каталоге пользователя и выполняются, когда пользователь загружает X-сервер из командной строки или входит в систему через экран регистрации программы XDM. Чтобы организовать вызов диспетчера окон или графической оболочки, необходимо добавить в сценарии ссылку на нужную программу. На с. 168 рассказывалось о том, как это сделать.

Но существуют и альтернативы программе XDM, например GDM (GNOME Desktop Manager) и KDM (KDE Display Manager). Эти программы позволяют выбирать графическую оболочку на этапе регистрации в системе. На рис. 13.6 изображено окно регистрации программы GDM. Интересующие нас команды находятся в меню Session. В частности, при выборе команды Xsession выполняется пользовательский сценарий `.xsession`, при выборе команды Gnome запускается оболочка GNOME, а при выборе команды Failsafe xterm запускается графическая оболочка в минимальной конфигурации. Программы GDM и XDM запоминают, какой выбор сделал пользователь, и автоматически подставляют нужный вариант при следующей регистрации. Дополнительные сведения о конфигурировании этих программ приведены в главе 21.

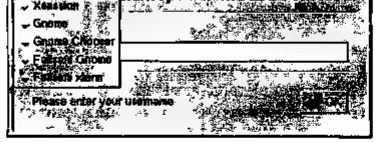


Рис. 13.6. Программы графической регистрации позволяют выбирать диспетчер окон или графическую оболочку

Скорость перемещения указателя мыши

Скорость перемещения указателя определяет расстояние, которое он проходит на экране в ответ на перемещение мыши. Этот параметр можно изменить с помощью

утилиты xset, которая часто вызывается в сценариях [illegible]. Ниже приведен пример соответствующей команды.

```
$ xset m 4
```

На заметку Эту команду нужно ввести в терминальном окне после запуска X-сервера, иначе будет выдано сообщение об ошибке.

После экспериментирования с утилитой xset m определите наиболее предпочтительную установку. Если используются методы регистрации, при которых выполняются сценарии .xsession или .xinitrc, включите в них приведенную команду. Она должна располагаться перед вызовом диспетчера окон.

Графические оболочки часто содержат удобные утилиты, позволяющие настраивать параметры мыши в графическом режиме. К примеру, на рис. 13.7 изображено окно программы GNOME Control Center, в котором активизирован раздел настройки мыши (выбирается из списка Peripherals в левой части). Пользоваться такой программой гораздо удобнее, чем утилитой xset, а кроме того, графическая оболочка сохраняет сделанные установки при перезагрузке.

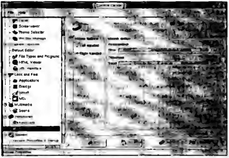


Рис. [illegible]. В графических оболочках есть программы, [illegible] параметров указателя мыши

Скорость повтора

Еще одной пользовательской настройкой является скорость ввода повторных символов при удержании клавиши. Этот параметр можно изменить несколькими способами.

- **Файл XF86Config.** Стандартную скорость повтора можно задать в разделе InputDevice (того, который относится к клавиатуре, а не к мыши) файла

XF86Config. Соответствующая опция называется AutoRepeat. Она принимает два аргумента: задержка перед началом повтора символа (в миллисекундах) и собственно скорость повтора (в секундах). В следующей строке задается повтор символа 10 раз в секунду после начальной задержки длительностью 250 мс:

```
Option "AutoRepeat"  250 10
```

- **Утилита xset**. То же самое способна сделать и утилита xset. Для этого нужно воспользоваться опцией r rate, указав начальную задержку и скорость повтора. Например, команда xset r rate 250 10 эквивалентна показанной выше строке конфигурационного файла. Пользователи могут включать эту команду в сценарии .xinitrc и .xsession.
- **Графические утилиты**. Программы наподобие GNOME Control Center, которые используются для задания параметров мыши, содержат и средства настройки параметров клавиатуры. Самое главное, что сделанные изменения сохраняются при перезагрузке.

Запуск нескольких X-сеансов

Интересной особенностью X Window System является возможность запуска нескольких X-сеансов на одном компьютере. Это позволяет работать одновременно в разных графических режимах, с разными диспетчерами окон или графическими оболочками, что особенно удобно для проверки программы на совместимость с разными оболочками.

На заметку

Не путайте X-сеансы с виртуальными рабочими столами, которые создаются многими диспетчерами окон и графическими оболочками. Все виртуальные рабочие столы имеют одинаковые разрешение и цветовую разрядность и функционируют в рамках единого серверного процесса, тогда как X-сеансы представляют собой отдельные процессы, конфигурация которых может отличаться.

Текстовые виртуальные терминалы позволяют запускать несколько программ в текстовом режиме и переключаться между ними по нажатию <Alt+F*n*>, где *n* — число от 1 до 8. X-сервер обычно работает на девятом виртуальном терминале, поэтому его можно вызвать из текстового режима, нажав <Alt+F9>. Для переключения на текстовый виртуальный терминал из графического режима нужно нажать <Ctrl+Alt+F*n*>.

Дополнительным X-сеансам соответствуют виртуальные терминалы с номерами 10 и выше. Главное — сообщить системе номер терминала, чтобы она запустила новый сеанс. Это делается такой командой:

```
$ startx -- :1 vt10
```

Ее нужно ввести в текстовом режиме. Если X-сервер уже работает, нажмите <Ctrl+Alt+F1>, чтобы переключиться в текстовый режим, зарегистрируйтесь и введите показанную команду. После короткой задержки должен начаться новый X-сеанс. Первому сеансу (номер 0) соответствует виртуальный терминал 9, а второму (номер 1, задаваемый параметром :1 в строке запуска) — виртуальный терминал 10. Всего может быть четыре сеанса — по числу свободных функциональных клавиш.

Как уже было сказано, каждому сеансу разрешается назначать свои параметры запуска. Предположим, стандартный сеанс работает в режиме 16-разрядной цветовой палитры. Тогда новый сеанс можно запустить в режиме 32-разрядной палитры, указав в командной строке опцию -bpp 32:

```
$ startx -  1  bpp 32 vt10
```

А как сделать, чтобы у каждого сеанса было другое разрешение экрана? Для этого нужно в разделе `Screen` файла `XF86Config` закрепить за каждой цветовой разрядностью определенное разрешение экрана. Тогда при выборе конкретной разрядности будет автоматически активизироваться и соответствующее разрешение. Есть и другой вариант — создать несколько версий файла `XF86Config` и выбирать нужную из них с помощью опции `-xf86config`:

```
$ startx -  :1 -xf86config ./XF86Config lowres vt10
```

На заметку Только суперпользователь имеет право указывать полное имя файла в опции `-xf86config`. Если рядовой пользователь создает нестандартный файл `XF86Config`, этот файл должен находиться в его начальном каталоге.

Резюме

Несмотря на то что в своей основе X Window System является довольно простой системой в сравнении со многими конкурирующими графическими средами, ее возможности велики. Простоту можно даже считать достоинством, ведь именно благодаря ей к системе подключаются произвольные графические интерфейсы и диспетчеры окон.

Глобальные конфигурационные параметры X-сервера, такие как разрешение экрана и файл устройства, предназначенный для взаимодействия с мышью, хранятся в файле `/etc/XF86Config`. Его можно редактировать непосредственно, но лучше пользоваться конфигурационными утилитами, которые позволяют быстро внести необходимые изменения. Пользовательские установки задаются в сценарии регистрации (`.xsession` или `.xinitrc`, в зависимости от того, как запускается X-сервер: автоматически или нет) или в специальных программах графических оболочек.

Полный справочник по FreeBSD

Часть III

Сетевое конфигурирование

Полный справочник по

Глава 14

Основы сетевого конфигурирования

Этой главой начинается третья часть книги посвященная сетевому конфигурированию. FreeBSD-системы широко используются в качестве сетевых серверов, поэтому указанная тема чрезвычайно важна для администраторов FreeBSD. Даже тем, кто использует систему дома, часто приходится заниматься этими вопросами при организации домашней сети или подключении к провайдеру Internet. В настоящей главе рассказывается о том, как настроить сетевые параметры системы. Последующие главы раздела посвящены коммутируемому доступу клиент-серверным настройкам и конфигурации встроенного фильтра пакетов

Сеть можно конфигурировать по-разному поэтому в первую очередь нужно изучить общие принципы организации сетей. Далее мы рассмотрим способ конфигурирования сети, традиционно применявшийся в UNIX. Он заключается в использовании статических IP-адресов. Современные сети можно конфигурировать так, чтобы центральный сервер динамически назначал IP-адреса другим компьютерам. Такие системы проще конфигурировать, и об этом тоже пойдет речь в главе. В завершение мы поговорим о тестировании и отладке сети Следует помнить, что некоторые из описываемых настроек являются временными Если необходимо, чтобы система автоматически включала соответствующие установки при перезагрузке, нужно внести определенные изменения в конфигурационные файлы

Локальные и коммутируемые сети

Прежде чем приступить к изучению сетевых возможностей FreeBSD, следует пояснить ряд ключевых принципов организации сетей Современные сети чаще всего реализуются на основе технологии *Ethernet*, имеющей несколько разновидностей В большинстве FreeBSD-систем данные передаются по сети с использованием протоколов *TCP/IP*. Это наиболее популярное семейство сетевых протоколов, лежащее в основе сети Internet. Последняя образована из множества сетей меньшего размера, соединенных посредством *маршрутизаторов*. Это усложняет задачу адресации компьютеров, которая решается за счет использования системы доменных имен — DNS.

Сетевое оборудование

Если нужно создать новую сеть, необходимо купить все ее аппаратные компоненты и подключить их надлежащим образом При добавлении компьютера к существующей сети изменению подвергается лишь небольшой участок сети В любом случае следует убедиться в совместимости всех компонентов, поскольку существуют различные стандарты сетевого оборудования В локальных сетях используется несколько типов устройств

- **Сетевые интерфейсы** Каждый компьютер, подключаемый к сети, должен иметь сетевой интерфейс. Обычно он добавляется с помощью сетевой платы, которая подключается к аппаратной шине (ISA или PCI) компьютера У некоторых компьютеров сетевой интерфейс встроен в материнскую плату Есть и другие виды интерфейсов. Сетевой интерфейс — это единственный аппаратный компонент сети для которого во FreeBSD требуется драйвер
- **Сетевые кабели** В локальных сетях компьютеры соединяются посредстаом кабелей В современных 100-мегабитовых сетях Ethernet используются кабели категории 5 (неэкранированная витая пара НВП) Они напоминают обычные телефонные кабели, но с более широкими разъемными соединениями Некоторые типы сетей создаются на основе оптоволоконных кабелей. Завоевывают

популярность также беспроводные технологии. В них не нужны кабели, так как данные передаются посредством радиоволн.

- **Концентраторы или коммутаторы.** В сетях определенного вида требуется центральное устройство, соединяющее различные компьютеры (если в сети всего два компьютера, то в этом нет необходимости). Одно из таких устройств называется *концентратором*. Это повторитель сигналов, передающий одни и те же данные всем подключенным к нему компьютерам. *Коммутатор* — более "интеллектуальное" устройство. Он передает пакет только тому компьютеру, которому он адресован. Коммутаторы Ethernet не намного дороже концентраторов, потому являются предпочтительным выбором. В сетях старого типа, например 10Base5 или 10Base2, ни концентраторы, ни коммутаторы не используются, а все компьютеры соединяются как бы в одну линию (это так называемая *шинная топология*, являющаяся противоположностью *звездообразной топологии*, формируемой концентраторами и коммутаторами).
- **Маршрутизаторы.** Для соединения двух небольших локальных сетей или подключения сети к Internet может использоваться маршрутизатор. FreeBSD-система умеет выполнять большинство функций маршрутизатора, но иногда без специализированных устройств не обойтись. Зачастую функции маршрутизации берет на себя провайдер Internet.

Большинство новых локальных сетей создается на основе 100-мегабитового Ethernet-оборудования с кабелями категории 5. В старых 10-мегабитовых сетях проложены либо кабели категории 5, либо толстые или тонкие коаксиальные кабели. Некоторые локальные сети создаются по технологии Token Ring. В них используются концентраторы, а скорость передачи данных составляет 16 Мбит/с (в новых версиях — 100 Мбит/с). Другие технологии применяются редко. Оптимальную производительность обеспечивают гигабитные сети. В них используются НВП или оптоволоконные кабели, а также гигабитовые коммутаторы. Это более дорогое оборудование, чем в 100-мегабитовых сетях, но цены неуклонно снижаются. В будущем ожидается широкое внедрение еще более производительных сетей.

Если FreeBSD-система подключается к существующей сети, необходимо выяснить ее тип. В случае 10- или 100-мегабитовой сети Ethernet не возникнет проблем с поиском сетевой платы, совместимой с FreeBSD. Список плат можно посмотреть в файле `HARDWARE.TXT`.

Протоколы TCP/IP

Сетевые протоколы — это "язык", на котором общаются компьютеры. Протоколы семейства TCP/IP не связаны с конкретным сетевым оборудованием. Они реализуются поверх Ethernet, Token Ring и других аппаратных интерфейсов. С помощью протокола PPP (Point-to-Point Protocol) протоколы TCP/IP применяются в последовательных и модемных соединениях (такая конфигурация описывается в главе 15). TCP/IP — это один из нескольких доступных *сетевых стеков*, причем наиболее популярный. Под сетевым стеком понимается набор протоколов, позволяющих компьютерам взаимодействовать по сети. Протоколы организованы иерархическим способом, отсюда представление о стаке. На вершине иерархии находятся пользовательские приложения и серверы, на нижнем уровне — драйверы и аппаратные устройства. Популярность TCP/IP обусловлена особенностями следующими протоколов.

- **Расширенное адресное пространство.** У каждого компьютера в сети должен быть адрес. Для этого в протоколах TCP/IP используются числовые адреса (называются *IP-адресами*). Разрядность такого адреса составляет 32 бит (четыре байта). Это означает, что всего существует 2^{32}, или 4294967296 адресов. На пер-

вый взгляд это огромное число, однако применявшиеся ранее схемы выделения адресов были очень неэффективными, поэтому вопрос нехватки адресов стоит очень остро. В протоколе IP следующего поколения — *IPv6* (текущая версия называется *IPv4*) — используются 128-разрядные адреса, что позволяет получить $3,4\times10^{38}$ адресов.

- **Имена компьютеров.** Людям часто удобнее работать не с числовыми, а с текстовыми адресами. В TCP/IP имеется механизм преобразования IP-адресов в *доменные имена*. Об этом пойдет речь на с. 307.
- **Поддержка множества портов.** На компьютере могут работать сотни сетевых программ. Каждая из них подключается к одному или нескольким сетевым *портам*. Это специальные виртуальные адреса в пределах компьютера, напоминающие внутренние телефонные номера в офисных АТС. Сетевые порты позволяют программам работать в сети одновременно и даже взаимодействовать с одной и той же удаленной системой. Данные автоматически распределяются таким образом, чтобы попадать к нужному адресату.
- **Маршрутизация.** Маршрутизаторы позволяют соединять не только компьютеры, но и целые сети. Наиболее крупной объединенной сетью является Internet. В других сетевых стеках средства маршрутизации очень ограничены, что снижает их ценность.
- **Межплатформенная поддержка.** Семейство TCP/IP изначально проектировалось для ранних UNIX-систем, но разработка велась открытым способом, что позволило со временем перенести протоколы в другие операционные системы. Современное семейство TCP/IP доступно на любой платформе, начиная от карманных компьютеров и заканчивая системами верхнего класса. Широкое распространение протоколов привело к появлению огромного количества сетевых приложений.

В большинстве сетевых стеков, включая TCP/IP, вводится понятие *клиента* и *сервера*. Клиент инициирует сетевое соединение, запрашивая данные, а сервер отвечает на запросы. Термины "клиент" и "сервер" могут относиться как к отдельным программам, так и к целым компьютерам. Из контекста обычно понятно, что означает соответствующий термин.

Другие сетевые стеки

Указанные выше особенности TCP/IP определяют превосходство этого семейства протоколов над другими сетевыми стеками. В основном альтернативные стеки являются локальными, т.е. обеспечивают работу приложений в локальных сетях. Ниже описаны наиболее распространенные стеки.

- **NetBEUI.** Это стек протоколов, используемых для организации совместного доступа к файлам и принтерам в среде Windows. Аналогичные функции можно реализовать и поверх TCP/IP (во FreeBSD это делается с помощью пакета Samba, который работает по протоколам TCP/IP, а не NetBEUI). NetBEUI не поддерживает маршрутизацию, а схема именования компьютеров является двухуровневой (используются имена компьютеров и рабочих групп или доменов).
- **AppleTalk.** Это аналог NetBEUI для Macintosh. Протоколы AppleTalk тоже используются для организации совместного доступа к файлам и принтерам в локальных сетях. В данном случае 32-разрядный адрес разбивается на две части; схема именования тоже двухуровневая. Протоколы AppleTalk в основном применяются лишь в компьютерах Apple. Во FreeBSD есть пакет Netatalk, реализующий протоколы AppleTalk поверх TCP/IP.

- **IPX** Это стек протоколов компании Novell разрабатывавшийся для объединения сетей, но не достигший такой гибкости как TCP/IP. В настоящее время он используется для организации совместного доступа к файлам и принтерам в сетях Novel NetWare.

Мы рассматриваем протоколы TCP/IP, потому что это наиболее популярный сетевой стек, значительно опережающий своих конкурентов. Другие семейства протоколов тоже могут использоваться во FreeBSD, хотя средства их поддержки ограничены

Мвршрутизация

Если FreeBSD-система работает в изолированной локальной сети (которая не подключена к Internet или другим сетям), то функции маршрутизации не нужны В остальных случаях без маршрутизации не обойтись.

Маршрутизатор соединяет две и более сети, в которых может использоваться различное сетевое оборудование. Например, маршрутизатор позволяет соединить локальную сеть Ethernet с локальной сетью Token Ring и связать их с внешним миром. Подобная конфигурация изображена на рис 14 1 В данном случае маршрутизатор является частью всех трех сетей.

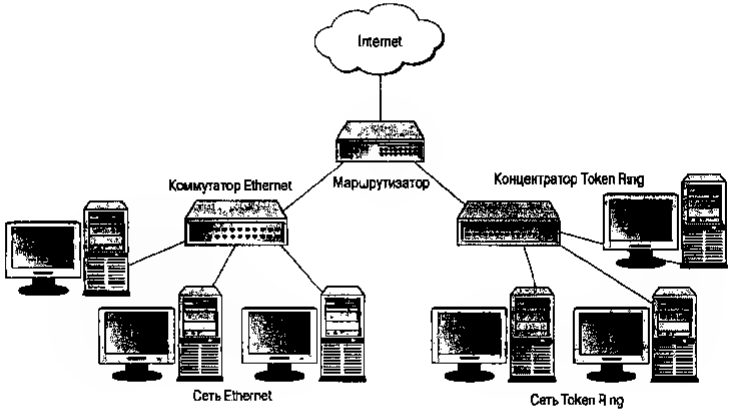


Рис 14 1. Маршрутизаторы ретранслируют данные между сетями TCP/IP даже если в этих сетях используется различное физическое оборудование

Предположим, что пользователь находящийся в сети Ethernet, хочет получить доступ к ресурсу, находящемуся в сети Token Ring. Такой доступ становится возможным благодаря маршрутизатору, через который проходит трафик между двумя сетями Точно так же маршрутизатор управляет доступом в Internet из обеих локальных сетей. Вообще говоря, в схеме, изображенной на рис. 14 1, присутствует множество маршрутизаторов, поскольку "облако" Internet представляет собой сложный набор взаимосвязанных маршрутизаторов и компьютеров, которые они обслуживают

Протоколы TCP/IP проектировались таким образом чтобы локальным компьютерам не требовалось "знать" детали маршрутизации за исключением IP адреса локального маршрутизатора. Компьютеры могут посылать данные непосредственно на другие компьютеры локальной сети Для этого достаточно указать адрес получателя и

оправить пакет через сетевой интерфейс. Получатель обязательно обнаружит такой пакет и обработает его. Что касается удаленных систем, то такой уровень доступа невозможен. В этом случае компьютер должен послать пакет локальному маршрутизатору, который "знает", что с ним делать дальше. Маршрутизатору, представленному на рис. 14 1, известны адреса локальных систем и адрес следующего маршрутизатора скрытого в "облаке" Internet. Пакет передается по цепочке маршрутизаторов, пока наконец, не достигнет адресата. Таким образом каждая система взаимодействует лишь со своими ближайшими соседями

С IP-адресами и маршрутизацией связано понятие *сетевой маски* (*или маски под сети*) Она представляет собой число, дополняющее IP-адрес и разбивающее его на адрес сети и адрес компьютера. Все компьютеры одной подсети имеют общий компонент IP-адреса. Это сетевая его часть. Остальные биты адреса уникальны для каждого компьютера и определяют машинную часть адреса. Сетевую маску можно представить в виде двоичного числа, в котором единичные биты относятся к сетевой части адреса, а нулевые — к машинной. Например, если IP-адрес компьютера равен 192.168.0.23, а сетевая маска — 255.255 255.0, то первые три байта (192.168.0) задают адрес сети, а последний байт (23) является адресом компьютера в подсети. Сетевую маску можно записать одним числом (называется *длиной маски*). Для этого нужно подсчитать количество следующих подряд единичных битов, начиная от старшего. В рассмотренном примере получим следующую спецификацию: 192.168.0.23/24 (маска записывается через косую черту). Без сетевой маски IP-адрес будет неполным В сетях, имеющих частные адреса в диапазоне 1292 168 *x.x*, сетевая маска обычно равна 255.255.255 0. Маршрутизаторы проверяют сетевую часть адреса и определяют, через какое сетевое соединение переслать пакет.

Граница между сетевой и машинной частями адреса традиционно проходила по границе байтов Это означает, что каждый байт маски был равен либо 255, либо 0 Все адреса разбивались на *классы*, за каждым из которых закреплялась своя маска. Адреса класса A (маска 255.0 0 0) соответствовали огромным сетям, адреса класса B (маска 255 255 0 0) — сетям среднего размера, адреса класса C (маска 255.255.255.0) — небольшим сетям. Классы D и E были зарезервированы для специального применения В настоящее время распределение на классы не играет особой роли С целью максимально эффективного использования адресного пространства IP-адреса выделяются блоками с маской произвольной длины Тем не менее конфигурационные утилиты предполагают наличие стандартной маски, если маска не указана явно.

FreeBSD-систему можно сконфигурировать в качестве маршрутизатора, но эта тема выходит за рамки книги (в главе 17 описывается, как включить фильтрацию пакетов для обеспечения безопасности системы) В большинстве случаев достаточно задать адрес существующего маршрутизатора Это может быть устройство, находящееся в локальной сети или на сервере провайдера Internet. Необходимо также сообщить компьютеру IP-адреса других компьютеров локальной сети Об этом будет рассказываться в следующем разделе

IP-адреса и доменные имена

В TCP/IP используются две схемы обозначения компьютеров Первая заключается в использовании IP-адресов Это фундаментальный способ адресации компьютеров. Вторая схема основана на текстовых именах компьютеров, которые проще запоминать пользователям

На заметку

На аппаратном уровне используются адреса третьего типа называемые MAC-адресами. В сетях Ethernet это 6 байтовое число присваиваемое каждой сетевой плате ее изготовителем. В стеке TCP/IP есть протоколы позволяющие узнать MAC-адрес по имеющемуся локальному IP-адресу

Имена компьютеров записываются в иерархическом виде а компоненты имени отделяются друг от друга точкой, например www.osborne com. Первый компонент обозначает *машинное имя* (в данном случае www), а остальные компоненты образуют *доменное имя* (osborne.com). Доменные имена выделяются организациям и физическим лицам, которые управляют назначением машинных имен в рамках домена. Владельцу домена разрешено создавать *поддомены*, представляющие собой логически независимые области родительского домена. К примеру администрация университета может создать факультетские поддомены, такие как english.pangaea.edu и physics.pangaea.edu. В рамках этих поддоменов имена компьютеров будут выглядеть так: byron.english.pangaea.edu и curie.physics.pangaea.edu.

Последний (самый правый) компонент имени обозначает домен верхнего уровня Такие домены разделяют сеть Internet по географическому или функциональному принципу. Например, домен com предназначен для коммерческого применения, тогда как учебные заведения сосредоточены в домене edu. Для каждой страны предусмотрен двухбуквенный домен верхнего уровня, например us для США или ru для России

Имена компьютеров и IP-адреса образуют *систему доменных имен* (Domain Name System, DNS), которая представляет собой распределенную базу данных. Обнаружив имя компьютера Internet-приложение посылает запрос локальному серверу DNS, пытаясь узнать соответствующий IP-адрес Поскольку DNS — распределенная система такой запрос может обрабатываться целой иерархией серверов, начиная от сервера верхнего уровня и заканчивая сервером поддомена. Для того чтобы это преобразование имело место, клиенты и серверы FreeBSD должны знать адрес DNS-сервера. В его отсутствие сетевым приложениям придется указывать IP адреса, что неудобно Адреса Web-узлов электронной почты и т.д записываются в виде доменых имен, по этому DNS — важный элемент глобальной сети

Использование статических IP-адресов

Каждый компьютер в сетях TCP/IP должен иметь IP-адрес FreeBSD-система должна знать не только свой собственный адрес, но и адрес маршрутизатора и DNS сервера Проще всего задать эти адреса вручную, как будет описано ниже. Другой подход заключается в использовании локального сераера, который будет назначать IP-адреса остальным компьютерам локальной сети. Эта конфигурация рассматривается на с. 314. При подключении компьютера к существующей сети выясните, какой способ выделения адресов применяется в ней. В небольшой сети проще работать со статическими адресами, по крайней мере первоначально.

Получение IP-адреса

В первую очередь нужно вмяснить IP-адрес системы Если компьютер включается в существующую сеть обратитесь к сетевому администратору. Если инсталлируется система, имеющая выход в Internet, то необходимую информацию может предоставить провайдер В небольших изолированных сетях используются IP-адреса из зарезервированных пулов (192.168 0.0—192.168 255 255, 172 16.0.0—172.31.255.255 и 10.0 0.0—10.255.255.255). IP-адрес не должен оканчиваться на .0 или .255, за исключением нескольких случаев. Помимо самого IP-адреса нужно получить еще сетевую маску. Если задать ее неправильно, сетевые функции будут выполняться но некоторые компьютеры окажутся недоступны

Важно узнать также IP-адрес маршрутизатора (его еще называют адресом *шлюза*) и адреса одного или нескольких DNS-серверов (многие провайдеры поддерживают резервный сервер) В изолированной локальной сети адрес маршрутизатора не исполь-

зуется, а DNS-сервер нужно организовать самостоятельно В небольшой сети проще вручную отредактировать файл `/etc/hosts` на каждом компьютере установив привязки между именами компьютеров и IP-адресами

Настройка сетевых параметров с помощью программы sysinstall

Статические IP-адреса, как и многие другие компоненты, можно конфигурировать с помощью программы `sysinstall` Следуйте приведенной ниже процедуре

1. Перейдите в режим суперпользователя и введите **`/stand/sysinstall`** для запуска программы `sysinstall`
2. Выберите в основном меню программы пункт `Configure`
3. Выберите в меню конфигурирования FreeBSD пункт `Networking`
4. Выберите в меню сетевых служб пункт `Interfaces`. Программа отобразит перечень сетевых интерфейсов (рис 14.2).
5. Выберите из списка сетевую плату (она, скорее всего, будет первой в списке) Учтите, что в список включены интерфейсы PPP и SLIP, используемые при работе с последовательными и модемными соединениями
6. Программа спросит, нужно ли настроить параметры протокола IPv6 Это довольно редкий протокол, поэтому мы его не рассматриваем. Выберите `No`.
7. Программа спросит, нужно ли использовать протокол DHCP для конфигурирования интерфейса В случае статической конфигурации выберите `No`
8. Далее программа отобразит форму, в которой следует ввести сетевые параметры системы (рис 14 2) В поле `IPv4 Address` указывается IP адрес системы, в поле `Netmask` сетевая маска, в поле `IPv4 Gateway` IP-адрес маршрутизатора, в поле `Name Server` — адрес DNS-сервера В небольших локальных сетях последние два поля можно оставлять пустыми В поле `Host` задается имя компьютера (достаточно ввести машинную часть имени программа `sysinstall` сама дополнит ее), а в поле `Domain` — доменное имя Если эти поля задать некорректно FreeBSD будет функционировать, но некоторые Internet-приложения не смогут нормально работать

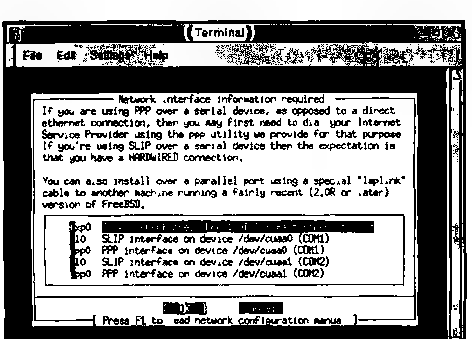


Рис 14 2. Программа `sysinstall` предлагает выбрать интерфейс

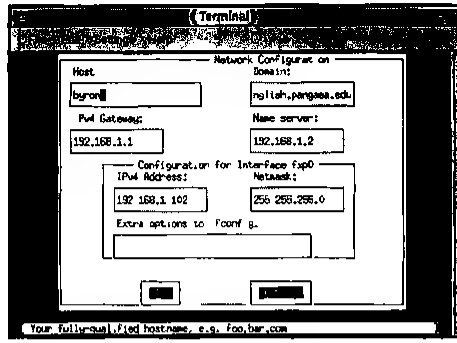


Рис. 14.3 Форма Network Configuration предназначена для задания параметров TCP/IP

На заметку Если провайдер или сетевой администратор предоставляет адреса нескольких DNS-серверов, введите любой из них или же только тот, который обозначен как первичный. На с. 312 будет показано, как задать дополнительные адреса. Это позволит продолжать работу в случае выхода из строя одного из серверов.

9 Программа спросит, нужно ли активизировать интерфейс немедленно. Выберите `Yes`.

С этого момента сетевой интерфейс должен функционировать нормально. В разделе "Тестирование конфигурации" рассказывается о том, как протестировать интерфейс.

Совет Конфигурационная информация добавляется в файл `/etc/rc.conf`, при этом старые записи не удаляются. Если удалить новые записи, изменения будут отменены.

Настройка адресов вручную

Иногда рассмотренные установки удобнее делать вручную, используя специальные утилиты или редактируя конфигурационные файлы напрямую. Это дает возможность настраивать множество специальных параметров, параллельно просматривая отладочную информацию.

Активизация интерфейса

При самостоятельном конфигурировании сетевого соединения нужно сначала активизировать сетевой интерфейс. Для этого предназначена команда `ifconfig`, синтаксис которой в упрощенном виде выглядит так:

```
ifconfig [интерфейс] [up   down] [адрес[/длина_маски]]
         [netmask маска]
```

Рассмотрим назначение аргументов команды:

- *интерфейс*. Этот аргумент задает имя сетевого интерфейса. Имена существенно отличаются в зависимости от модели сетевой платы. Чтобы узнать нужное

имя, просмотрите сообщения ядра сразу после загрузки системы. Для этого введите **dmesg | less** и поищите строку, в которой идентифицируется сетевое устройство К примеру, в следующей строке Ethernet-плата обозначена как fxp0:

```
fxp0  Ethernet address 00 03:47.b1.ee:b8
```

- **up | down** Опция up предназначена для включения интерфейса, а опция down — для отключения По умолчанию предполагается первый вариант, по этому опцию up можно опускать
- *адрес*[*/длина_маски*] Этот аргумент задает IP адрес интерфейса Через ко сую черту допускается указывать длину сетевой маски
- **netmask** *маска* Маску можно задать и в явном виде воспользовавшись опци ей netmask. Если маска вообще отсутствует, команда ifconfig самостоятель но вычисляет ее на основании класса IP-адреса.

При отсутствии аргументов команда ifconfig выдает отчет по всем сетевым ин терфейсам Если задать имя интерфейса, будет получен отчет по нему, например:

```
$ ifconfig fxp0
fxp0  flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet 192.168.1.6 netmask 0xffffff00 broadcast 192 168.1 25
        ether 00:03:47·b1 ee·b8
        media: Ethernet autoselect (100baseTX
        status. active
```

Наиболее важная информация здесь это IP адрес (приведен в строке inet). Раз он указан, значит, компьютер обрабатывает трафик, направляемый по этому адресу

Наличие IP-адреса позволяет взаимодействовать с другими компьютерами локаль ной сети Например, можно выполнить команду ping (рассматривается на с 315) и получить ответ от локального компьютера

Настройка маршрутизации

Две команды связаны с конфигурированием протоколов маршрутизации Первая из них это netstat, которая выводит сведения о текущей конфигурации сети В частности, опция -r включает вывод таблицы маршрутизации (воспользуйтесь также опцией -n, чтобы команда отображала IP-адреса, а не имена компьютеров, особенно если механизм преобразования имен не работает) Рассмотрим пример

```
$ netstat -rn
Routing tables

Internet:
Destination   Gateway           Flags    Refs     Use  Netif  Expire
default       192 168.1.1       UGSc        0      13   fxp0
127.0.0 1     127.0.0 1         UH          0      48    lo0
192 168 1     link#1            UC          2       0   fxp0
192 168.1 1   0:a0.cc:.4·ba:2    UHLW        3      44   fxp0    1186
192 168.1.3   0 50:bf·19.7e·99   UHLW        2     719   fxp0     591
```

Эта команда сообщает много полезной информации Проанализируем ее

- **Локальные системы** В последних двух строках примера приводятся адреса сис тем, находящихся в той же подсети (192 168 1.1 и 192.168 1 3) В колонке Gateway для них указан 6-байтовый MAC-адрес. FreeBSD удалит эти записи по истечении заданного числа секунд (колонка Expire), но их можно сгенери ровать повторно Трафик по этим адресам проходит через интерфейс fxp0 (колонка netif)

- **Локальный сетевой адрес**. Строка, помеченная как `Link#1` в поле `Gateway`, идентифицирует локальный сетевой адрес. Трафик по этому адресу проходит через интерфейс `fxp0`. Для определения MAC-адреса целевой системы FreeBSD использует низкоуровневые протоколы стека TCP/IP, после чего добавляет запись для нее в таблицу маршрутизации. Локальный сетевой адрес определяется автоматически, когда команда `ifconfig` активизирует сетевой интерфейс.
- **Адрес обратной связи**. Во FreeBSD существует виртуальный сетевой интерфейс `localhost` с адресом 127.0.0.1. Это так называемый *интерфейс обратной связи*, его адрес соответствует самому компьютеру. Такой интерфейс используется некоторыми низкоуровневыми утилитами.
- **Стандартный маршрут**. Стандартный маршрут определяет получателя трафика, для которого нет более конкретных записей в таблице маршрутизации. В рассмотренном примере это строка, начинающаяся с ключевого слова `default`. В данном случае стандартным маршрутизатором является узел 192.168.1.1.

У только что включенного сетевого интерфейса стандартный маршрут скорее всего не задан. Чтобы добавить его, воспользуйтесь комаидои `route` следующего вида:

```
# route add default 192.168.1.1
```

Если введен неправильный маршрут, его можно удалить, указав опцию `delete` вместо `add`. При удалении нестандартного маршрута опустите ключевое слово `default`.

С этого момента можно пользоваться командой `ping` и другими сетевыми утилитами для доступа к локальным и удаленным системам по IP-адресам. Далее необходимо настроить механизм распознавания имен.

Задание имени компьютера

Команда `hostname` позволяет назначить компьютеру имя. При отсутствии аргументов команда сообщает текущее имя, в противном случае - модифицирует его. Вот пример:

```
# hostname byron.english.pangaea.edu
```

Имя компьютера используется некоторыми почтовыми клиентами при отправке почты. Кроме того, почтовый сервер может ограничивать прием почты в зависимости от имени компьютера.

Наличие у компьютера локального имени не означает, что его можно использовать для адресации компьютера в Internet. Эта функция возлагается на DNS-серверы. Чтобы система стала доступной по определенному имени, свяжитесь с администрацией домена, к которому относится имя, и зарегистрируйте его.

Настройка DNS

Если DNS-сервер доступен, то достаточно отредактировать конфигурационный файл `/etc/resolv.conf` (листинг 14.1). Он содержит IP-адреса от одного до трех DNS-серверов. Здесь же могут быть заданы доменное имя компьютера и список доменов, которые разрешается просматривать. Это позволяет пользоваться короткими машинными именами вместо полных имен.

Листинг 14.1. Образец файла /etc/resolv conf

```
domain english.pangaea edu
search classics.pangaea edu pangaea edu
nameserver 192.168 1 2
nameserver 10.40 91.4
```

Компьютер, сконфигурированный в соответствии с листингом 14 1, просматривает сначала домен english.pangaea.edu, а затем — домены classics.pangaea.edu и pangaea.edu. Запросы посылаются серверам с адресами 192 168.1.2 и 10.40.91 4 В принципе строка search не нужна, так как она может привести к замедлению поиска. Предположим, пользователь ввел **telnet shakspere**, исказив имя компьютера. Прежде чем выдать сообщение об ошибке, FreeBSD придется послать запросы к доменам shakspere.english.pangaea.edu, shakspere.classics.pangaea.edu и shakspere.pangaea.edu На выполнение каждого такого запроса уходит время поэтому сообщение об ошибке может появиться через полминуты. Более того, если вдруг окажется, что существует компьютер shakspere.pangaea.edu, пользователь начнет сеанс доступа не к тому компьютеру

Альтернативой обычному DNS-поиску является использование файла /etc/hosts. Изначально в нем содержатся одна или две адресные привязки, в частности для узла localhost (127.0 0.1). Записи файла имеют следующий синтаксис:

```
IP-адрес    полное имя    сокращенное_имя
```

В последнем поле обычно задается машинное имя Рассмотрим образец записи

```
192 168 1.43  shakespeare.english.pangaea.edu shakespeare
```

В случае такой конфигурации пользователь сможет получить доступ к компьютеру, даже если DNS-сервер не отвечает или не задан Это довольно эффективный способ преобразования имен в небольших сетях где нет необходимости запускать DNS сервер

После того как механизм преобразования имен сконфигурирован (адреса DNS-серверов указаны в файле /etc/resolv.conf), к локальным и удаленным компьютерам можно обращаться как по IP-адресам так и по именам

Закрепление изменений

Установки, задаваемые командами ifconfig и route, действуют лишь до момента перезагрузки системы Чтобы восстановить их, придется вводить команды повторно (конфигурационные параметры DNS хранятся в файлах, поэтому о них можно не беспокоиться).

Если необходимо сделать конфигурационные изменения постоянными, добавьте соответствующие команды в сценарии запуска системы С сетевым конфигурированием связан один сценарий. /etc/rc.conf. Он может содержать строки следующего вида

```
ifconfig fxp0="inet 192 168.1 102 netmask 255 255 255 0"
defaultrouter="192.168.1 1"
hostname="byron.english.pangaea.edu"
```

Эти строки не совсем идентичны командам ifconfig, route и hostname, но в них содержится та же информация Именно этот файл модифицирует программа sysinstall при конфигурировании параметров сети Обычно программа добавляет записи к файлу, не удаляя существующие. Более поздние записи имеют приоритет Это позволяет легко отменять изменения, удаляя ненужные строки или превращая их в комментарии

Протокол DHCP

Конфигурирование компьютера на использование статических IP адресов не представляет трудностей, особенно если делать это с помощью программы наподобие `sysinstall`. Однако статическая конфигурация имеет ряд недостатков. Для сетевого администратора настройка IP-адреса каждого компьютера вручную может обернуться настоящим кошмаром в условиях частого изменения конфигурации сети. Поэтому были разработаны различные методы автоматического назначения IP-адресов. В локальных сетях чаще всего используется протокол DHCP (Dynamic Host Configuration Protocol — протокол динамического конфигурирования компьютеров).

На заметку В сетях DSL (Digital Subscriber Line — цифровая абонентская линия) применяется другой протокол, PPPoE (PPP over Ethernet). Он рассматривается в главе 15.

В некоторых конфигурациях требуется, чтобы MAC адреса DHCP клиентов были зарегистрированы в базе данных DHCP-сервера. Если это так, сетевой администратор или провайдер запросит соответствующую информацию. MAC-адрес (Ethernet-адрес) — это 6-байтовое число, обычно записываемое в шестнадцатеричном виде. Если оно не указано в документации к сетевой плате, воспользуйтесь командой `ifconfig`, как было показано выше. Можно также просмотреть сообщения утилиты `dmesg` сразу после загрузки системы. Вот как это делается:

```
$ dmesg | grep "Ethernet address"
fxp0: Ethernet address 00:03:47:b1:ee:b8
```

DHCP-сервер ожидает поступления специальных пакетов из локального сетевого сегмента. Это *широковещательные* пакеты, т.е. они направляются всем компьютерам данного сегмента. В ответ на широковещательный пакет сервер сообщает клиенту его конфигурационные данные, включая IP-адрес, сетевую маску, адрес шлюза, адреса DNS-серверов и, возможно, машинное и доменное имена компьютера. DHCP-сервер может каждый раз назначать компьютеру один и тот же адрес либо периодически менять его (это может произойти только после перезагрузки, поскольку без уважительной причины IP-адрес не меняется).

Если используется программа `sysinstall`, то порядок настройки системы будет таким же, как и в случае статической конфигурации (см. с. 309). Только на запрос по поводу протокола DHCP нужно ответить `Yes`. В результате система осуществит поиск DHCP сервера в локальной сети, после чего отобразит форму `Network Configuration` (рис. 14.3). Поля формы можно оставить пустыми.

Теперь FreeBSD будет запрашивать IP-адрес у сервера с помощью утилиты `dhclient`. Соответствующая установка хранится в файле `/etc/rc.conf` и выглядит примерно так.

```
ifconfig_fxp0="DHCP"
```

Эту строку можно добавить в файл самостоятельно, без помощи программы `sysinstall`. После перезагрузки система запросит свои сетевые параметры у DHCP-сервера.

Тестирование конфигурации

После того как сетевые параметры системы сконфигурированы, ее нужно протестировать. Своевременное тестирование позволяет оперативно локализовать источник проблем.

Базовая проверка с помощью команды ping

Основное средство тестирования сети — команда `ping`. Она посылает целевой системе небольшой служебный пакет, в ответ на который должен вернуться другой служебный пакет. Подобный обмен пакетами происходит на нижних уровнях стека TCP/IP и не зависит от конфигурации высокоуровневых средств таких как Web-серверы или клиенты. Это позволяет проверить функционирование стека TCP/IP а также наличие соединения с целевой системой.

В большинстве случаев достаточно указать имя или IP-адрес опрашиваемой системы. В первую очередь нужно проверить IP адреса локальных компьютеров, например:

```
% ping 192.168 1 3
PING 192.168.1 3  192.168.1.3). 56 data bytes
64 bytes from 192 168.1.3: icmp_seq=0 ttl=255 time=0.335 ms
64 bytes from 192.168.1.3: icmp_seq=1 ttl=255 time=0.326 ms
64 bytes from 192.168.1.3: icmp_seq=2 ttl=255 time=0.318 ms
64 bytes from 192.168.1.3: icmp_seq=3 ttl=255 time=0.312 ms
^C
- - 192.168 1.3 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.3120/0.323/0 335,0.009 ms
```

Если не задана опция `-c` *счетчик*, команда посылает тестовые пакеты до тех пор пока пользователь не нажмет <Ctrl+C>. Как видно из примера, команда сообщает о том, что на каждый отправленный пакет получен ответ При этом указано время ожидания В конце выдается статистический отчет с указанием общего числа отправленных и полученных пакетов и параметров ожидания. В локальной сети потерь пакетов обычно не бывает (можно правда, прервать работу команды до того, как она успеет получить ответ) Время ожидания непродолжительное не более одной миллисекунды

Попробуйте в качестве начального объекта тестирования использовать локальный маршрутизатор Чтобы сеть функционировала, он должен быть доступен. Предварительно необходимо узнать IP-адрес маршрутизатора с помощью команды `netstat -nr`.

Если начальная проверка завершилась неудачно, проверьте IP-адрес и сетевую маску собственного компьютера. Возможно, они заданы неправильно. Введите команду `ifconfig`, чтобы определить конфигурацию сетевого интерфейса. Если он неактивен, воспользуйтесь опцией `up` При наличии DHCP-сервера поищите в журнальном файле (`/var/log/messages`) сообщения об ошибках, выдаваемые утилитой `dhclient` Например, в файле могут присутствовать такие записи.

```
Apr  7 22:02:25 hairloprillalar dhclient: Can't bind to dhcp%
address  Address already in use
Apr  7 22:02 25 hairloprillalar dhclient  exiting.
```

Сообщение `Address already in use` говорит о том, что какая-то другая программа захватила порт DHCP-сервера Возможно, утилита `dhclient` уже выполняется, а вы пытаетесь ее переконфигурировать Уничтожьте утилиту и перезапустите ее

Некоторые компьютеры сконфигурированы так, чтобы не реагировать на запросы команды `ping` Следовательно, ошибка тестирования не всегда означает ошибку конфигурации Попробуйте несколько способов проверки

Причиной неудачного выполнения команды `ping` могут быть и аппаратные проблемы. разрыв кабеля, отключение коммутатора и т.д Проверьте целостность кабеля поправьте разъем, подключенный к сетевой плате, или протестируйте другой компьютер, который гарантированно работает правильно с тем же сетевым оборудованием.

Проверка преобразоввния имен

Если базовая команда ping выполняется успешно, а в системе сконфигурирован DHCP-сервер или используется файл /etc/hosts, попробуйте вместо IP адреса за дать имя компьютера. Команда должна работать точно так же как и в предыдущем случае Если вы обратили внимание, в первой строке вывода команды IP-адрес указан дважды. При использовании имени компьютера первый из этих адресов будет заме нен именем. Когда механизм преобразования имен не функционирует, эта строка от сутствует Если же он работает, но целевая система недоступна после вывода первой строки команда "зависнет".

На заметку Если DNS-сервер не находится в локальной сети, то ошибка обраще ния DNS может быть вызвана проблемами с маршрутизацией

Попробуйте проверить, распознаются ли имена компьютеров, находящихся в Internet. Если DNS-сервер расположен в локальной сети, то преобразование имен может выполняться даже в случае недоступности удаленного узла

Правильность преобразования имен можно также проверить с помощью команд host и nslookup. Первая из них возвращает IP адрес, соответствующий заданному имени, а вторая — имя, соответствующее IP-адресу

Выявление ошибок маршрутизации

С помощью команды ping можно проверить наличие соединения с удаленным компьютером Рассмотрим следующий пример

```
% ping www.osborne.com
PING www.osborne.com (198.45 24.130): 56 data bytes
64 bytes from 198.45 24.130: icmp_seq=0 ttl=238 time=131.518 ms
64 bytes from 198.45.24.130: icmp_seq=1 ttl=238 time=162.523 ms
64 bytes from 198.45 24.130: icmp_seq=2 ttl=238 time=130.061 ms
64 bytes from 198.45.24.130: icmp_seq=3 ttl=238 time=134.463 ms
^C
--- www.osborne.com ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 130.061/139.641/162.523/13.306 ms
```

Как показывает проверка, время ожидания здесь гораздо больше, чем в локальной сети В общем случае оно зависит от типа соединения, расстояния до целевой систе мы и других факторов Если команда не выполняется, то, возможно, проблема не в маршрутизации а в DNS. Повторите запрос, указав вместо имени компьютера IP адрес

Если одни внешние узлы доступны, а другие — нет то проблема может заключать ся в неправильно заданной сетевой маске Не исключено также, что не работает внешний маршрутизатор или сама целевая система Чтобы проверить это, воспользуй тесь командой traceroute, которая отображает путь пакета от одной системы к дру гой Рассмотрим следующий пример

```
$ traceroute -n 68.1.0 44
traceroute to 68.1.0.44 (68.1.0 44), 64 hops max  40 byte packets
1   192 168.1.1  1.299 ms  1.185 ms  1.165 ms
2   10 1.96.1  16.236 ms  46.519 ms  68.663 ms
3   68.9.8.81  41.946 ms  10 701 ms  47.766 ms
4   68.9.14.5  37.191 ms  14.907 ms  45.332 ms
5   68 1 0.44  36.135 ms *  45.391 ms
```

Опция n заставляет команду не выполнять преобразование имен, что ускоряет ее работу Каждый маршрутизатор опрашивается три раза с помощью служебных паке тов, поэтому команда сообщает три времени ожидания. В этом примере первый мар-

шрутизатор является локальным а следовательно, отвечает быстрее всех Последую щие маршрутизаторы реагируют гораздо медленнее Целевая система ответила только на два пакета из трех (звездочка указывает на отсутствие ответа). Возможно, она работает некорректно. Похожие проблемы могут возникать и на промежуточных маршрутизаторах В этом случае остается только ждать когда провайдер устранит источник проблемы

Утилита `whois` может выдать информацию о том, кто отвечает за функционирование компьютера Для этого нужно указать ей IP-адрес или имя искомой системы Обычно сообщается контактная информа ция иапример адрес электронной почты или номер телефона.

Резюме

Для современных компьютеров корректная сетевая конфигурация чрезвычайно важна Это особенно справедливо в отношении FreeBSD-систем, которые часто используются в качестве сетевых серверов или рабочих станций, включенных в сеть. Программа `sysinstall` позволяет настроить компьютер на использование статического IP-адреса или на получение адреса у DHCP-сервера. Завершив конфигурирование, следует убедиться в правильности установок. Для этого существуют различные утилиты основная из которых — `ping`

Полный справочник по

Глава 15

Коммутируемые сети

В главе 14 рассказывалось о том, как сконфигурировать FreeBSD в локальной сети Ethernet или Token Ring. Но многие компьютеры, особенно домашние, имеют сетевые соединения другого типа: они подключены к Internet по протоколу PPP (Point-to-Point Protocol), который обеспечивает функционирование протоколов TCP/IP в модемных и других последовательных каналах. Компьютер, расположенный в изолированной локальной сети, тоже может иметь коммутируемое телефонное подключение к Internet. В этой главе рассматриваются принципы конфигурирования таких соединений во FreeBSD. Будет также описана разновидность протокола PPP, называемая PPPoE (PPP over Ethernet). Этот протокол применяется многими Internet-провайдерами, даже несмотря на наличие у них Ethernet-оборудования.

Тестирование модема

Прежде чем приступать к конфигурированию Internet-соединения, необходимо протестировать работу модема во FreeBSD. Современные модемы бывают трех типов:

- **Модемы с последовательным интерфейсом RS-232.** Такие модемы подключаются к последовательному порту компьютера либо представляют собой плату расширения ISA или PCI, на которой установлен встроенный последовательный порт. Они почти всегда работают во FreeBSD, лишь иногда может потребоваться изменить номер запроса на прерывание (IRQ) или отключить последовательный порт материнской платы. Доступ к этим устройствам осуществляется через файлы `/dev/cuaa0`–`/dev/cuaa3`.
- **Внутренние бесконтроллерные модемы.** Большинство внутренних модемов *бесконтроллерные*. Это означает, что в них отсутствуют многие традиционные аппаратные компоненты, функции которых переложены на специальные драйверы. Их еще называют *программными модемами* (семейство *WinModems*). Такие модемы обычно не работают во FreeBSD. Для некоторых из них (особенно для популярных моделей Lucent) имеются драйверы, но их трудно найти. На момент написания книги в коллекции переносимых приложений существовал раздел `comms/ltmdm`, однако ссылки в нем были недействительными.
- **USB-модемы.** Внешние модемы все чаще оснащают интерфейсом USB (Universal Serial Bus), а не RS-232. Но на сегодняшний день поддержка USB-модемов во FreeBSD 5.0 пока еще ограничена.

Как нетрудно понять из этого списка, лучше всего работать с внешними модемами, имеющими последовательный интерфейс. Впрочем, старые внутренние модемы тоже поддерживаются достаточно хорошо.

Чтобы протестировать основные функции модема, подключите его к компьютеру, включите питание и воспользуйтесь терминальной программой, например текстовой утилитой `minicom` или графической программой Seyon (обе они могут быть инсталлированы в виде пакета или переносимого приложения). Например, для запуска программы Seyon нужно ввести `seyon -modem /dev/cuaa0` (подставьте реальное имя файла устройства). Далее наберите команду вида `ATI`*число*, чтобы получить статусную информацию от модема. Команда `ATDT###-####` заставляет модем набрать указанный телефонный номер (вместо `#` подставляется цифра телефонного номера). Описание команд модема должно быть приведено в руководстве пользователя. На рис. 15.1 изображено окно программы Seyon с отчетом об успешном сеансе подключения к модему.

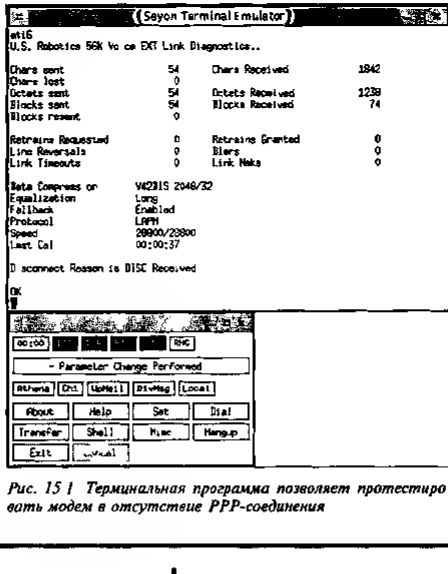


Рис. 15.1. Терминальная программа позволяет протестировать модем в отсутствие PPP-соединения

Использование графического номеронабирателя

Новичкам удобнее устанавливать соединение с помощью графической утилиты набора номера. После соответствующего конфигурирования пользователю остается лишь выбрать нужный номер телефона из списка и щелкнуть на кнопке Connect. Программа набора номера обычно настраивается таким образом, чтобы соединения могли устанавливаться рядовыми пользователями. Но программами данного класса нельзя управлять посредством внешних сценариев, поэтому они не подходят, если, к примеру, система должна автоматически подключаться к Internet в обозначенное время (скажем, для регулярной доставки почты). В этом случае используются сценарии набора номера, описанные в следующем разделе.

Выбор номеронабирателя

Существует множество утилит набора номера, работающих во FreeBSD. Если инсталлирована оболочка KDE, то в системе должен быть номеронабиратель KPPP. Для оболочки GNOME аналогичная программа называется GNOME-PPP, но она не инсталлируется в стандартном варианте оболочки и к тому же (на момент написания книги) недоступна в формате переносимого приложения. По этой причине для рассмотрения была выбрана программа X-ISP (`http://xisp.hellug.gr`). Она работает в любой графической оболочке и доступна как в виде пакета, так и в формате переносимого приложения.

Создание соединения с провайдером

Чтобы иметь возможность подключаться к серверу провайдера, необходимо выполнить две конфигурационные настройки. В первую очередь следует создать *файл аутентификации*, в котором хранятся имя пользователя и его пароль на сервере. Во-вторых, нужно сообщить программе X-ISP номер телефона, по которому сервер принимает запросы, скорость соединения и т.д

На заметку Некоторые номеронабиратели позволяют конфигурировать файл аутентификации непосредственно из окна подключения, задавая имя пользователя и пароль в соответствующих полях. Программа X-ISP требует чтобы этот файл редактировался вручную.

Создание файла аутентификации

FreeBSD (точнее демон pppd, отвечающий за установление PPP-соединения) хранит аутентификационную информацию в файле `/etc/ppp/pap-secrets`. Этот файл предполагает использование аутентификации по протоколу PAP (Password Authentication Protocol - протокол парольной аутентификации), который наиболее распространен в настоящее время Но некоторые провайдеры по-прежнему применяют CHAP (Challenge Handshake Authentication Protocol — протокол аутентификации с предварительным квитированием) В этом случае необходимо создать файл `/etc/ppp/chap-secrets` который имеет точно такие же формат и содержимое, как и файл `pap-secrets`

На заметку Раньше провайдеры часто применяли процедуры регистрации, в которых требовалось, чтобы пользователь написал сценарий, посылающий регистрационное имя и пароль в ответ на приглашения сервера Сего дня подобные конфигурации встречаются редко, но программа X ISP поддерживает такой метод регистрации Если он используется файл аутентификации создавать не нужно

Записи файла аутентификации имеют следующий формат

имя клиента имя сервера пароль [IP адрес[...]]

Под именем клиента понимается имя пользователя, предоставляемое провайдером (оно может отличаться от регистрационного имени FreeBSD) Имя сервера обычно неизвестно, поэтому во втором поле чаще всего стоит звездочка (*), указывающая на то, что допустимо любое значение. В поле пароля задается пароль, используемый для регистрации на сервере провайдера, а не во FreeBSD Наконец, можно указать список IP-адресов, принимаемых от провайдера Как правило, этот список пуст, поскольку большинство провайдеров назначает IP-адреса динамически

Внимание В файле аутентификации содержится незашифрованный пароль. Убедитесь в том, что этот файл доступен только суперпользователю (его владельцем должен быть пользователь `root`, режим доступа — 0600) Указанный пароль должен применяться *исключительно* для подключения к серверу провайдера, а не для регистрации на любом другом компьютере, включая текущую систему Это позволит свести к минимуму риск хранения незашифрованного пароля

Можно создавать записи для нескольких пользователей или провайдеров В этом случае программа или сценарий набора номера предложат выбрать требуемое имя пользователя. Ниже показаны записи пользователей `agb` и `alex`, каждая со своим паролем:

```
agb   *   w8tS4nr6
alex  *   b3l1gm!
```

Конфигурирование программы X-ISP

Программу X-ISP разрешается запускать пользователю root и членам группы dialer. Наличие специальной группы позволяет администратору контролировать, кто имеет право работать с программой. О том как добавлять пользователей в группы рассказывалось в главе 10.

Чтобы вызвать программу, введите **xisp** в терминальном окне. В результате появится основное окно программы (рис 15 2) Ниже указан порядок действий, которые нужно выполнить для конфигурирования программы

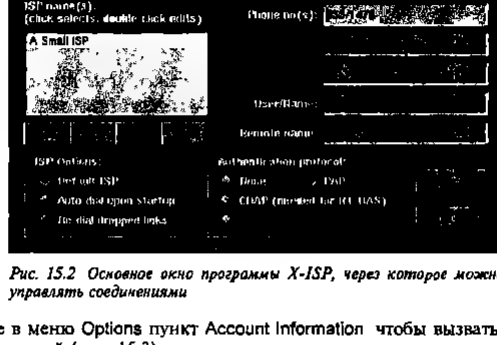


Рис. 15.2 Основное окно программы X-ISP, через которое можно управлять соединениями

1 Выберите в меню Options пункт Account Information чтобы вызвать окно настроек учетных записей (рис. 15 3)

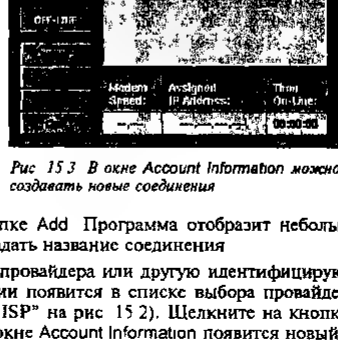


Рис 15 3 В окне Account Information можно создавать новые соединения

2 Щелкните на кнопке Add Программа отобразит небольшое диалоговое окно в котором следует задать название соединения

3 Введите название провайдера или другую идентифицирующую информацию Эта строка впоследствии появится в списке выбора провайдера в главном окне программы ("A Small ISP" на рис 15 2). Щелкните на кнопке OK, после чего в списке ISP name(s) в окне Account Information появится новый элемент

4. Введите номер телефона провайдера в поле Phone no(s). Код города указывать не нужно. Запятая между цифрами означает паузу в наборе номера. Она может быть необходима при подключении к внешней телефонной линии
5. Выберите переключатель PAP или CHAP в группе Authentication protocol, чтобы задействовать соответствующий аутентификационный протокол. Если выбрать переключатель None, придется написать собственный сценарий набора номера
6. Введите имя пользователя в поля User/Name и Remote name. В зависимости от конфигурации провайдера заполнению может подлежать одно из этих полей или оба.
7. Щелкните на кнопке OK в окне Account Information.
8. Выберите в меню Options пункт Connection Options чтобы вызвать окно настроек соединения (рис. 15.4).

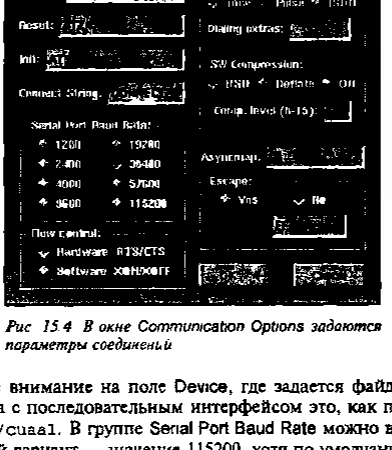


Рис 15.4 В окне Communication Options задаются параметры соединений

9. Обратите особое внимание на поле Device, где задается файл модемного устройства Для модема с последовательным интерфейсом это, как правило, файл `/dev/cuaa0` или `/dev/cuaa1`. В группе Serial Port Baud Rate можно выбрать скорость модема Наилучший вариант — значение 115200, хотя по умолчанию задано 38400
10. Выберите в меню Options пункт TCP/IP Options чтобы вызвать окно настроек TCP/IP (рис 15 5).
11. В окне TCP/IP Options задаются различные параметры, связанные с конфигурацией стека TCP/IP. Большинство из этих параметров описывалось в главе 14. Как правило, менять их не требуется, за исключением установок группы DNS support Нужно получить у провайдера IP-адреса одного или двух DNS-серверов, выбрать переключатель Yes и задать эти адреса в полях Primary DNS Server и Secondary DNS Server. Система может запрашивать адреса автоматически в процессе конфигурирования протокола PPP, но они не всегда передаются корректно.

Теперь программа X-ISP готова устанавливать PPP-соединения, по крайней мере теоретически Лишь в редких случаях требуется более тонкая настройка

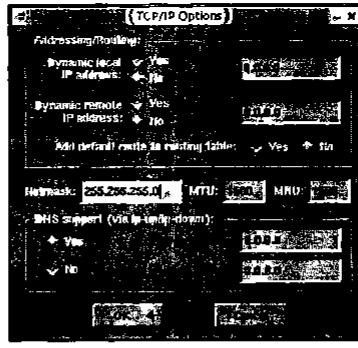


Рис 15.5 Окно TCP/IP Options

Программа X ISP устанавливает соединения с помощью демона pppd, который записывает журнальные сообщения в файл /var/log/messages. Если соединение не удалось установить, просмотрите этот файл, выполнив команду **tail -n 20 /var/log/messages**

Установление соединения

Чтобы установить соединение, выберите нужного провайдера из списка и щелкните на кнопке Connect в главном окне программы X-ISP (рис. 15.2) После этого вы услышите, как модем набирает номер (при условии, что включен динамик модема), а в главном окне появится информация о ходе установления соединения Затем в поле Modem Speed программа сообщит о том, какова скорость соединения, и через несколько секунд соединение будет окончательно установлено.

Для проверки соединения можно воспользоваться любой сетевой программой, например ping (описывалась в главе 14) Что касается Web-броузеров, почтовых клиентов и связанных с ними программ, то их требуется переконфигурировать, по крайней мере первоначально, указав адреса серверов провайдера Об этом будет рассказываться в главе 24 Можно также выполнить команду ifconfig чтобы проверить наличие интерфейса ppp0

```
$ ifconfig ppp0
ppp0. flags=8051<UP,POINTOPOINT,RJNNING,MULTICAST> mtu 1500
        inet 192.168 1.205 --> 192.168 1 2 netmask 0xffffff00
```

Если последняя строка (в ней указываются IP-адреса локального и удаленного компьютера) не отображается, значит, что-то сконфигурировано неправильно или между компьютерами нет стабильного соединения.

По окончании работы щелкните на кнопке Disconnect в главном окне программы X ISP Если используется внешний модем, он издает характерный щелчок и часть индикаторных лампочек гаснет, а в окне программы появляется сообщение "PPP Link is Down"

Использование текстовых утилит набора номера

Графические номеронабиратели удобны для пользователей, но они не всегда являются оптимальным средством. Текстовые утилиты набора номера не требуют наличия X-среды и к тому же позволяют создавать периодические задания, которые автоматически устанавливают соединение и выполняют необходимые операции без участия пользователя, например загружают электронную почту. Все это достигается путем непосредственного взаимодействия с демоном pppd. Поддержка протокола PPP обычно включена в ядре по умолчанию.

На заметку Есть также пользовательская PPP-утилита, называющаяся ppp. Ее конфигурационные установки хранятся в файле /etc/ppp/ppp.conf. Эта утилита рассматривается в разделе "Применение протокола PPPoE в сетях DSL", хотя ее использование не ограничено протоколом PPPoE.

Вспомогательные файлы

Прежде чем конфигурировать пользовательские или низкоуровневые PPP-утилиты, необходимо проверить ряд вспомогательных файлов. Один из них — файл аутентификации (pap-secrets или chap-secrets, в зависимости от того, какой протокол — PAP или CHAP — используется). Он описывался в предыдущем разделе.

Другой важный файл называется /etc/resolv.conf. В нем содержится список DNS-серверов провайдера. Этот файл описывался на с. 312. Некоторые провайдеры конфигурируют свои входные серверы так, чтобы информация о DNS предоставлялась пользователям автоматически, но это не всегда происходит корректно, поэтому приходится вручную заносить адреса DNS-серверов.

Конфигурирование протокола PPP

Конфигурационные установки протокола PPP хранятся в двух файлах. Первый из них — это файл опций, /etc/ppp/options. Второй — *сценарий набора номера*. В нем задаются строки, посылаемые демоном pppd модему, и строки, которые предполагается получить в ответ от модема. Наличие такого сценария позволяет демону набирать телефонный номер и в случае необходимости вводить команды для регистрации на сервере провайдера.

Файл /etc/ppp/options

Файл /etc/ppp/options содержит большинство опций демона pppd за исключением аутентификационных данных (опции можно передавать демону в виде аргументов командной строки). В листинге 15.1 показан типичный файл options.

Листинг 15.1. Образец файла /etc/ppp/options

```
/dev/cuaa0 115200
crtscts
modem
noipdefault
silent
defaultroute
user agb
connect "/usr/bin/chat -f /etc/ppp/chat.script
```

Рассмотрим назначение строк файла и других полезных опций

- **/dev/cuaa0** *скорость*. Нужно сообщить демону pppd через какой файл устройства следует устанавливать соединение. Обычно это файл /dev/cuaa0 или /dev/cuaa1, т.е первый либо второй последовательный порт Здесь же задается скорость работы порта Для современных модемов лучше всего подходит значение 115200
- **crtscts** — заставляет систему вести *аппаратный контроль передачи данных*. В этом случае обе стороны соединенил будут передавать через последовательный кабель сигналы о готовности принимать и передавать данные Такой режим необходим для обеспечения надежной работы высокоскоростных модемов
- **modem**. Демон pppd позволяет установить PPP-соединение с другим компьютером либо напрямую, либо используя модем как промежуточное устройство Во втором случае необходима опция modem, которая заставляет демон дожидаться сигнала обнаружения несущей и выполнять другие модемные операции
- **noipdefault**. Эта опция нужна в большинстве конфигураций. Она заставляет демон pppd принять IP-адрес, назначенный удаленной системой.
- **silent** — определяет процесс организации PPP-канала. Она обычно необходима в коммутируемых соединениях и сообщает демону pppd о том, что противоположная сторона должна первой выслать пакет LCP (Link Control Protocol — протокол управления каналом). Некоторые провайдеры требуют чтобы первый пакет посылался вызывающей стороной В таких случаях опцию нужно удалить
- **defaultroute** — заставляет демон pppd создать запись в таблице маршрутизации, указывая сервер провайдера в качестве стандартного шлюза Она нужна в коммутируемых PPP-соединениях.
- **user** *имя* — задает имя пользователя указанное в первом поле файла аутентификации
- **connect** *сценарий* — задает команду (берется в кавычки), инициирующую соединение. Обычно здесь содержится ссылка на сценарий, который набирает телефонный номер и, в случае необходимости регистрируется на сервере провайдера и выполняет команду инициализацнн PPP-сеанса Сценарий, вызываемый в листинге 15.1, работает в большинстве систем (его код приведен в листинге 15.2). При желании можно написать более сложный сценарий, который загружает электронную почту иди запускает сетевые утилиты.
- **disconnect** *сценарий* — аналог опции connect, но задает команду или сценарий, выполняемые по завершении соединения. Опцию disconnect можно использовать для завершения программ, запущенных сценарием подключения
- **connect-max-attempts** *n* - указывает, сколько раз демон pppd должен пытаться установить соединение Она полезна, если входные линии провайдера часто бывают заняты
- **debug** — опция, используемая при отладке. Она заставляет демон pppd записывать более подробные сообщения в файл /var/log/messages
- **demand** — заставляет демон pppd устанавливать соединение только при наличии исходящего трафика
- **idle** *n* — заставляет демон pppd разрывать соединение, если оно неактивно в течение указанного числа секунд. Такая мера предосторожности позволяет предусмотреть ситуацию, когда пользователь ушел оставив компьютер подключенным к Internet Опция idle должна присутствовать если указана опция demand

Демон pppd принимает много других опций. Все они описаны на соответствующей man-странице. В большинстве случаев конфигурация показанная в листинге 15.1, оказывается вполне работоспособной

Сценарий набора номера

Сценарий /etc/ppp/chat-script, ссылка на который содержится в листинге 15.1, управляет подключением к серверу. Как правило, он сбрасывает состояние модема, набирает заданный телефонный номер и дожидается установления модемного соединения (это не то же самое, что PPP-соединение, которое устанавливается на более высоком уровне) Типичный сценарий набора номера показан в листинге 15.2 Он анализируется программой /usr/bin/chat Сценарий состоит из конфигурационных установок, специфичных для программы chat, а также набора строк, которые программа chat ожидает от модема и посылает в ответ. Формально все аргументы программы chat записываются в одну строку, но, чтобы упростить редактирование файла, список аргументов представляют в виде таблицы в конце каждой строки которой стоит обратная косая черта

Листинг 15.2 Образец файла /etc/ppp/chat-script

```
TIMEOUT    30 \
ABORT      BUSY \
ABORT       NO CARRIER' \
''         ATZ \
OK         ATDT555 1475 \
CONNECT
```

На момент написания книги программа chat содержала ошибку которая не позволяла ей корректно интерпретировать строки разделенные подобным образом. Поэтому листинг 15.2 нужно преобразовать в однострочное представление Возможно со временем эта ошибка будет устранена

Первые три строки листинга 15.2 сообщают программе chat о том, когда следует разрывать соединение. В частности, на завершение каждой команды отведено 30 секунд Если в ответ на одну из команд получено сообщение BUSY или NO CARRIER, работа прекращается, поскольку модему не удалось установить соединение. Добавьте другие строки ABORT, чтобы обрабатывать дополнительные сообщения об ошибках.

Последние три строки определяют параметры дозвона. Первая из них заставляет программу chat послать команду ATZ в ответ на любые входные данные. Эта команда восстанавливает начальное состояние модема Следующая строка заставляет программу послать команду ATDT555-1475 (тоновый набор номера 555-1475) в ответ на приглашение OK, которое генерируется модемами после получения команды ATZ. Последняя строка означает что программа должна дождаться приглашения CONNECT, ни чего не послав в ответ На этом выполнение сценария завершается и управление передается демону pppd

Программа chat после каждой строки посылает символ возврата каретки. Если нужно явно указать такой символ, воспользуйтесь управляющей последовательностью \r

Если провайдер не применяет протокол PAP или CHAP листинг 15 2 нужно расширить, включив обработку регистрационного диалога Добавьте к последней строке листинга обратную косую черту а затем введите следующие строки:

```
ogin.      agb \
ssword:    w8tS4nr6 \
rompt:     start-ppp
```

Эти строки подсказывают программе chat, как регистрироваться на сервере, который выдает приглашения login: и password:. Получив приглашение prompt:, программа посылает команду start-ppp. Естественно, имя пользователя, пароль и команда начала сеанса будут разными в зависимости от провайдера и учетной записи. Первые несколько символов в ожидаемых приглашениях не указаны, так как они иногда теряются в процессе передачи или имеют другой регистр.

Сценарий набора номера часто приходится модифицировать с учетом особенностей модема и провайдера. Это самая сложная задача, связанная с конфигурированием демона pppd. Воспользуйтесь терминальной программой, такой как minicom или Seyon, чтобы проанализировать сеанс подключения к провайдеру и выяснить, какие команды следует включить в сценарий.

Управление соединением

Когда все необходимые файлы сконфигурированы, соединение устанавливается очень просто: достаточно ввести pppd. Демон pppd принадлежит и доступен для выполнения пользователю root и группе dialer. После установления соединения в системе появится устройство ppp0.

```
# ifconfig ppp0
ppp0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
        inet 192.168.1.205   > 192.168.1.2 netmask 0xffffff00
```

Если вторая строка (в ней указываются IP-адреса локального и удаленного компьютера) не отображается, значит, соединение не функционирует. Подождите несколько секунд и попробуйте еще раз. Иногда демону pppd требуется время, чтобы все закончить. Если по истечении минуты ничего не изменилось, поищите в файле /var/log/messages сообщения, объясняющие причину ошибки. В случае необходимости включите в файл /etc/ppp/options опцию debug, чтобы увеличить количество журнальных сообщений, выдаваемых демоном pppd.

Когда соединение активно, становится возможен запуск любых сетевых программ, от ping до Mozilla. Полезные сведения по тестированию соединения приводились на с. 314. Чтобы разорвать соединение, необходимо уничтожить демон pppd.

```
# killall pppd
```

Если было активизировано несколько PPP-сеансов, определите идентификатор процесса того сеанса, который нужно уничтожить, и воспользуйтесь командой kill.

Применение протокола PPPoE в сетях DSL

Существует разновидность протокола PPP, называемая PPPoE (PPP over Ethernet). Этот протокол применяется в широкополосных соединениях, которые конфигурируются не так, как традиционные коммутируемые соединения.

Вместо того чтобы применять протокол PPPoE, можно приобрести широкополосный маршрутизатор, который самостоятельно взаимодействует с сервером провайдера по протоколу PPPoE, используя обычные протоколы TCP/IP для обмена данными с локальными системами. Тогда FreeBSD-системы будут конфигурироваться так, как описывалось в главе 14. Обзор нескольких широкополосных маршрутизаторов можно найти по адресу http://www.practicallynetworked.com/reviews.

Назначение протокола PPPoE

Протокол PPPoE позволяет провайдерам, предоставляющим широкополосный доступ, назначать IP-адреса клиентским компьютерам. В настоящее время протокол применяется преимущественно для обслуживания DSL-соединений домашних пользователей. Провайдеры кабельных сетей в основном используют протокол DHCP, а в высокоскоростных DSL-соединениях IP-адреса назначаются статически. В PPPoE инкапсулированные PPP-пакеты передаются через Ethernet-кабель, а не привычный последовательный кабель RS-232.

Зачем все-таки необходим протокол PPPoE? В конце концов, протоколы TCP/IP уже долгое время успешно функционируют в сетях Ethernet, а протокол PPPoE лишь все усложняет, добавляя еще один уровень инкапсуляции. Дело в том, что протокол PPPoE упрощает задачу провайдерам, помогая им управлять IP-адресами и контролировать использование учетных записей. Для конечных пользователей разница несущественна.

Программы для работы по протоколу PPPoE

Существуют две основные программы, позволяющие работать по протоколу PPPoE во FreeBSD:

- **ppp**. Та же самая пользовательская утилита ppp, которая реализует поддержку протокола PPP в последовательных соединениях, может применяться и для работы по протоколу PPPoE.
- **Roaring Penguin**. Пакет Roaring Penguin (`http://www.roaringpenguin.com/pppoe/`) был написан для Linux, но перенесен и во FreeBSD. Его достоинством является графический интерфейс.

Ниже будет описано применение стандартной утилиты ppp.

Конфигурирование протокола PPPoE

Конфигурационные параметры протокола PPPoE задаются в файле `/etc/ppp/ppp.conf` (листинг 15.3).

Листинг 15.3. Образец файла `/etc/ppp/ppp.conf`

```
default:
     set log Phase tun command
     set ifaddr 10 0.0.1/0 10.0.0.2 0

название провайдера:
     set device "PPPoE:fxp0"
     set authname agb
     set authkey w8tS4nr6
     set dial
     set login
     add default HISADDR
```

Некоторые установки файла требуют отдельного пояснения.

- *название провайдера*. В этой строке задается название конфигурации. Если утилита ppp используется для установления как широкополосных так и обычных коммутируемых соединений, создайте отдельный раздел для телефонного соединения (в стандартном файле `ppp.conf` такой раздел есть и называется `papchap`).
- **`set device PPPoE:fxp0`**. Эта строка сообщает утилите ppp о том, что будет использоваться протокол PPPoE, и заодно идентифицирует Ethernet-устройство. Вместо `fxp0` нужно подставить имя реального устройства, присут-

ствующего в системе (введите `ifconfig` чтобы получить список доступных устройств). Некоторые провайдеры требуют указания *тега службы*, который идентифицирует PPPoE-сервер. Это выглядит следующим образом: `set device PPPoE:`*устройство*`:`*тег*

- **Имя пользователя и пароль**. В строках `set authname` и `set authkey` задаются имя пользователя и его пароль на сервере провайдера. В листинге 15.3 они равны соответственно, `agb` и `w8tS4nr6`.
- **Стандартный маршрут**. Строка `add default HISADDR` заставляет утилиту `ppp` добавить стандартный маршрут для этого соединения. Если PPPoE-соединение является вторичным каналом, строку можно удалить, но в большинстве конфигураций она нужна.

Управление соединением

После того как протокол PPPoE сконфигурирован, установить соединение можно следующей командой:

```
# ppp -dial название_провайдера
```

Команда `ppp` выполняет процедуру обнаружения PPPoE-устройств, после чего организует канал связи с сервером провайдера. Если все пройдет успешно, пользователь получит доступ в Internet. С точки зрения пользователя, PPPoE-соединения ничем не отличаются от других Internet-соединений. Чтобы разорвать такое соединение, требуется уничтожить процесс, который его инициировал:

```
# killall ppp
```

Если необходимо, чтобы FreeBSD активизировала PPPoE-интерфейс на этапе начальной загрузки, добавьте следующие строки в файл `/etc/rc.conf`:

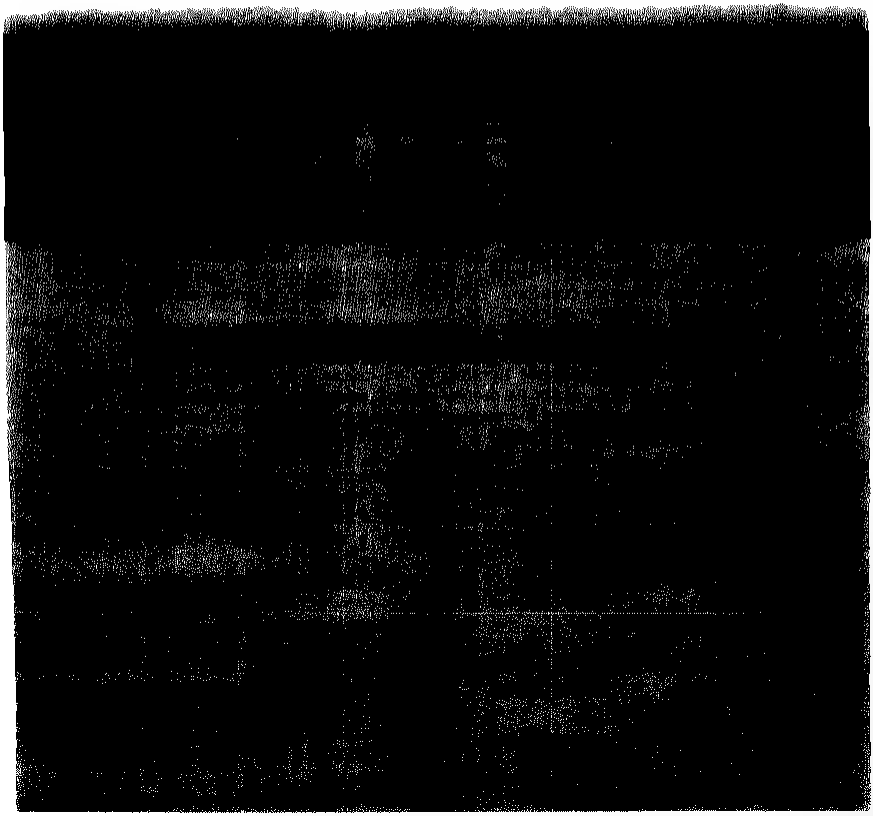
```
ppp_enable="YES"
    ppp_mode="ddial"
    ppp_nat="NO"
    ppp_profile="название_провайдера"
```

Название провайдера обязано соответствовать названию раздела в файле `/etc/ppp/ppp.conf`. Если система должна функционировать в качестве маршрутизатора, реализующего систему NAT (Network Address Translation), задайте параметр `ppp_nat` равным `YES`. Такой маршрутизатор позволяет множеству локальных компьютеров иметь один и тот же внешний IP-адрес. В упоминавшихся выше широкополосных маршрутизаторах система NAT реализована (подробнее об этом типе маршрутизации рассказывается в главе 17).

Резюме

В наши дни большинство компьютеров имеет временный или постоянный доступ в Internet. FreeBSD-системы чаще используются в качестве постоянно действующих серверов, но коммутируемые соединения остаются важными для домашних пользователей и небольших компаний. Такие соединения обычно устанавливаются по протоколу PPP, который поддерживается во FreeBSD. Для этого могут использоваться графические утилиты набора номера, например X-ISP, или текстовые утилиты наподобие `pppd`. В последнее время все большее распространение получает разновидность протокола PPP, называемая PPPoE. Этот протокол функционирует поверх Ethernet, а не в последовательных или модемных каналах. Его используют многие провайдеры, предоставляющие услуги DSL домашним пользователям. Во FreeBSD поддержка протокола PPPoE реализуется утилитой `ppp` или PPPoE-клиентом Roaring Penguin.

Полный справочник по



В главе 14 говорилось о том, что программы, работающие по протоколам TCP/IP могут быть клиентами и серверами. Клиенты инициируют сетевые операции, а серверы отвечают на запросы клиентов. В настоящей главе эта тема рассматривается подробно. Для наглядности будет проанализирован пример типичной сетевой транзакции. Мы поговорим о том, как запускать серверы во FreeBSD и рассмотрим ряд приемов сетевой диагностики.

Задачи клиентов и серверов

В любой сетевой транзакции одна из сторон начинает "говорить" первой. Это клиент. Он запрашивает передачу данных с сервера. Обычно в ходе последующего обмена клиент и сервер передают множество пакетов данных. Предварительно могут выполняться процедуры аутентификации, когда между клиентом и сервером устанавливается защищенное, зашифрованное соединение. Основной единицей обмена является файл, например файл, хранящийся на FTP-узле, или Web-страница. Почтовые сообщения тоже хранятся в виде файлов. Но не во всех сетевых операциях участвуют файлы. В операциях DNS-поиска передаются фрагменты информации, хранящейся в файлах описания доменов. В сеансах дистанционной регистрации происходит передача данных, вводимых пользователем и отображаемых программами.

Потоки данных могут идти в любом направлении. К примеру, в типичном HTTP сеансе Web-броузер (клиент) запрашивает файлы у сервера, поэтому большая часть данных идет от сервера к клиенту. Но в то же время пользователь FTP может записывать файлы на сервер и в SMTP-сеансе (Simple Mail Transfer Protocol — простой протокол передачи почты) почта отправляется с клиента на сервер.

Термины "клиент" и "сервер" относятся как к отдельным программам, так и к компьютерам, на которых эти программы работают. Назначение термина становится ясным из контекста. Иногда для ясности говорится "серверный компьютер". На компьютере могут одновременно функционировать как клиентские, так и серверные программы, например сервер Telnet, с которым пользователи работают в дистанционном режиме, и почтовые клиенты. Бывает, что одна и та же программа играет обе роли. Скажем, почтовые SMTP-серверы часто ретранслируют почту других серверов. Такая программа является сервером, когда принимает почту, и клиентом, когда посылает ее в другую систему.

Многие считают, что серверы — это высокопроизводительные компьютеры, находящиеся в отдельных помещениях, а клиенты — это компьютеры отдельных пользователей. Обычно это недалеко от истины, но на самом деле разделение на клиентов и серверы связано с тем, какая из сторон первой начинает передавать данные. Особой группой программ являются *X-серверы* (описывались в главе 13). Они реализуют графический интерфейс для FreeBSD и других UNIX-систем и подобно другим серверам, реагируют на запросы, поступающие от клиентских программ. При этом X-серверы управляют взаимодействием с пользователем: принимают данные от клавиатуры и мыши, отображают окна на экране и т.д. Они работают на компьютере, к которому подключены устройства ввода-вывода, а клиентские программы выполняются на компьютере, который может находиться в другой комнате и даже на другом конце земного шара.

Порты и их применение

Предположим, проектируется стек сетевых протоколов. Необходимо все организовать так, чтобы на одном компьютере могло работать несколько клиентов и серверов. Для этого клиент должен иметь возможность обращаться к конкретной серверной программе, а сервер должен отвечать на запросы того клиента, который к нему обратился. Например, почта, поступающая от одного клиента, не должна смешиваться с почтой других клиентов. Эту задачу берут на себя *порты* TCP/IP, которые представляют собой числовые идентификаторы, закрепленные за конкретными программами. В сочетании с IP-адресами порты уникальным образом идентифицируют каждую сторону сетевого соединения.

Номера портов

Когда сетевая программа обращается к стеку TCP/IP, ей назначается порт. Порты нумеруются от 0 до 65535 (порт 0 обычно не используется). Некоторые из них уже закреплены за стандартными серверами, другие доступны для любой программы, которая запрашивает порт. Программы могут запрашивать доступ к порту с конкретным номером или работать с выделенным операционной системой произвольным портом. Первый вариант характерен для серверов, второй — для клиентов. Серверы закрепляются за определенными портами, чтобы клиентам было проще к ним обращаться. Клиенту не нужен порт с конкретным номером; он просто сообщает номер назначенного ему порта серверу, чтобы тот мог направлять ответные пакеты.

На заметку

Порты TCP/IP отличаются от сетевых аппаратных портов. Последние представляют собой разъемы, куда вставляются Ethernet-кабели. Порты TCP/IP являются виртуальными, и в одном Ethernet-соединении может использоваться весь диапазон виртуальных портов.

Только одной программе разрешается работать с конкретным портом. Например, если Web-сервер закреплен за портом 80, ни одна другая программа не сможет подключиться к нему. Любые данные, адресуемые порту 80 компьютера, будут направляться Web-серверу. Когда Web-сервер отправляет данные, они помечаются как исходящие из порта 80 (некоторые программы открывают несколько портов; в таком случае программа сама указывает, какой порт следует использовать для передачи данных). Порты можно представить как внутренние номера телефонов в офисной АТС, позволяющие распределять звонки между внутренними телефонами компании.

По умолчанию FreeBSD назначает порты с номерами от 1024 и выше программам, которым не требуются конкретные порты. Порты с номерами 1023 и ниже являются *привилегированными*, т.е. доступ к ним разрешен только суперпользователю. Такое разграничение сделано из соображений безопасности. Серверы обычно взаимодействуют с привилегированными портами, чтобы их работе не могли помешать злоумышленники. К примеру, рядовому пользователю недоступен порт Web-сервера. В некоторых протоколах эта особенность используется как мера защиты при обработке клиентских запросов. Если клиент делает запрос с привилегированного порта, ему предоставляются более широкие привилегии на том основании, что такое соединение скорее всего было установлено системным администратором. К сожалению, в наши дни это не является надежной защитой, поскольку злоумышленники могут контролировать весь компьютер, в том числе и его привилегированные порты.

Стандартное распределение портов

Чтобы подключиться к серверу, сетевой клиент должен знать номер порта, на котором работает сервер. Разработчики стека TCP/IP закрепили целый ряд портов за наиболее распространенными серверами, что существенно упрощает их поиск. Например, Web-серверу всегда соответствует порт 80, почтовому SMTP-серверу — порт 25, серверу Telnet — порт 23. Клиенты "знают" об этом и делают правильные запросы, даже когда номер порта не указан. Если допустим ввести `telnet jeeves.threeroomco.com`, программа `telnet` подключится к порту 23. Некоторые клиенты позволяют переопределять стандартные установки. Например, если ввести `telnet jeeves.threeroomco.com 80` программа `telnet` свяжется с портом Web-сервера. Это удобно при отладке, так как программа `telnet` разрешает вводить команды других протоколов.

Стандартное распределение портов допускается переопределять. Это может понадобиться для запуска нескольких однотипных серверов на разных портах, обхода блокировок конкретных портов или запуска сервера от имени рядового пользователя на непривилегированном порту.

Во FreeBSD есть файл `/etc/services`, в котором заданы стандартные номера портов. В каждой записи файла указаны сетевой протокол, закрепленный за ним номер порта и тип порта. Например, следующая запись соответствует порту Telnet:

```
telnet          23/tcp
```

Привязки номеров портов к именам, сделанные в файле `/etc/services`, используются во многих серверных конфигурационных файлах. В частности, суперсерверы (описаны на с. 339) работают с именами служб, такими как `telnet` в рассмотренном примере. Эти привязки определяются общепринятыми стандартами, хотя в случае менее популярных серверов могут использоваться произвольные имена. На Web-узле `http://www.iana.org/assignments/port-numbers` приведен более полный вариант файла `/etc/services`, чем тот, что имеется во FreeBSD. Файл `/etc/services` разрешается редактировать, добавляя в него новые привязки. Удаление неиспользуемых привязок оправданно с точки зрения безопасности, так как это помешает серверам несанкционированно просматривающим файл `/etc/services`.

Порты TCP, UDP и других протоколов

В стеке TCP/IP предусмотрено несколько способов передачи данных. Два наиболее распространенных из них — с помощью TCP (Transmission Control Protocol — протокол управления передачей) и с помощью UDP (User Datagram Protocol — протокол передачи дейтаграмм пользователя). При использовании TCP между компьютерами формируется двусторонний канал связи с контролем ошибок. Этот протокол предпочтителен, когда необходимо надежное соединение для передачи большого числа пакетов. Его недостатком являются высокие расходы ресурсов на обслуживание каждого пакета, связанные с контролем ошибок и другими механизмами. При использовании UDP стабильный двусторонний канал не создается. Это довольно простой протокол, предназначенный для быстрой доставки небольших объемов данных. Например, механизм преобразования имен в DNS основан на использовании UDP, поскольку для выполнения задачи требуется всего несколько пакетов. UDP — высокоскоростной протокол, однако его нельзя считать надежным, поскольку доставка пакетов не гарантируется.

Когда речь шла о номерах портов, ничего не говорилось о различиях между пакетами TCP, UDP и других протоколов. Но если просмотреть файл `/etc/services`, в нем обнаружится множество ссылок на порты TCP и UDP. Теоретически за любым портом можно закрепить и протокол TCP-типа, и протокол UDP-типа. На практике такая несогласованность встречается крайне редко. Большинству протоколов назнача

ется один и тот же номер порта как для TCP-, так и для UDP режима передачи данных Это позволяет избежать путаницы, которая возникает в ситуации, когда на одном и том же порту работает два разных сервера

Помимо TCP и UDP в семейство TCP/IP входят и другие низкоуровневые протоколы, например ICMP (Internet Control Message Protocol — протокол управляющих сообщений в сети Internet) Для большинства из них нет привязок в файле /etc/services, потому что в этих протоколах отсутствует понятие порта. Протокол ICMP, к примеру, предназначен для передачи низкоуровневых сетевых сообщений Существуют различные типы сообщений, но необходимости в выделении отдельного порта нет, так как программы редко работают с пакетами этого протокола

Структура сетевого соединения

Чтобы лучше понять, как функционирует сеть, полезно рассмотреть порядок взаимодействия сервера и клиента и узнать, какие компьютеры и протоколы при этом задействуются. Естественно, реальное взаимодействие является более сложным, и в нем могут участвовать протоколы, не упомянутые здесь. В некоторых операциях происходит квитирование, т е взаимное согласование последующих действий, но это остается незамеченным пользователями В конце раздела рассказывается о важных расширениях стандартной модели взаимодействия.

Схема рассматриваемого примера приведена на рис. 16.1. Для простоты предполагается что клиентские и серверные сети основаны на технологии Ethernet

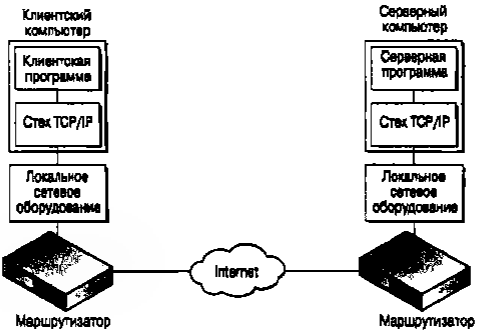


Рис 16 1 При сетевом обмене данными происходит взаимодействие программ на клиентском и серверном компьютерах с учетом различного сетевого оборудования

Устновлпвние соединения

Все начинается с клиентской программы которая получила указание передать данные Такое указание может поступить в режиме реального времени от пользователя или быть следствием автоматизированной операции. Клиентская программа направляет в стек TCP/IP локальной системы запрос, в котором указаны IP-адрес и порт удаленного сервера Протоколы стека осуществляют последовательную инкапсу

ляцию данных подготавливая их к передаче по сетевому кабелю В зависимости от уровня протокола информационные наборы могут называться *пакетами, фреймами* или как-то иначе Предположим, анализируется прохождение Web-трафика. В таком случае первоначально посылаются пакеты запрашивающие открытие TCP-соединения и передачу конкретного файла с Web-сервера

В большинстве случаев клиентской программе известно доменное имя компьютера Чтобы преобразовать его в IP адрес, программа должна обменяться данными с DNS-сервером. Такой обмен может быть достаточно сложным.

Некоторые сетевые запросы столь велики что их приходится разбивать на несколько пакетов. Сетевая подсистема берет эту задачу на себя, помечая каждый исходящий пакет, чтобы принимающая система могла восстановить исходный запрос.

Сетевая подсистема формирует из пакетов данных фреймы Ethernet, которые кодируются сетевой платой С этого момента данные готовы к передаче по сети. Сетевое оборудование включает в себя кабели, коммутаторы, концентраторы и другие устройства. На физическом уровне содержимое фреймов не меняется Данные просто ретранслируются от одного устройства к другому

Маршрутизация

На рис 16.1 изображены два маршрутизатора: один из них подключен к локальной сети, другой — к серверной сети Если клиент и сервер находятся в одной и той же сети, маршрутизатор не нужен, поскольку сервер обнаруживает пакеты, адресованные непосредственно ему. Если же данные поступают на маршрутизатор, он частично декодирует их В частности, маршрутизатор извлекает пакет TCP/IP из фрейма Ethernet, чтобы определить IP-адрес получателя Затем маршрутизатор повторно инкапсулирует данные, создавая новый фрейм, который, как правило, передается через другой сетевой интерфейс Иногда маршрутизатор вынужден разбивать фреймы, если их размер оказывается слишком велик для смежной сети Когда это происходит, последующие маршрутизаторы или непосредственно система-получатель должны выполнять сборку фрагментов.

При передаче пакета через Internet в дело вступает множество маршрутизаторов В каждом случае маршрутизатор удаляет "оболочку", созданную своим предшественником, а затем снова упаковывает данные для отправки их следующему маршрутизатору Когда данные наконец достигают маршрутизатора, подключенного к локальной сети сервера, он "понимает", что они адресованы в локальную сеть и потому формирует фрейм Ethernet, который передается серверному компьютеру при участии соответствующего локального сетевого оборудования

Ответ сервера

Сетевая подсистема сервера обнаруживает пакет, поступивший от локального маршрутизатора и начинает распаковку данных. Сначала удаляется заголовок Ethernet-фрейма, а затем данные других протоколов стека TCP/IP В итоге Web-сервер получает запрос на установление соединения и передачу данных

Сервер может ответить на запрос или проигнорировать его. Последнее может произойти из соображений безопасности — например, от внешней системы поступил запрос к службе, которая доступна только локальным пользователям. Предположим, что в нашем примере у Web-сервера нет причин игнорировать запрос, поэтому он подготавливает ответ. Сервер извлекает исходный IP-адрес и номер порта из клиентского запроса формирует требуемый документ и высылает его клиенту Данные передаются

по сети точно так же, как и в случае клиентского запроса только в обратном направлении Отличие может быть лишь в том, что данные пройдут через другой набор маршрутизаторов Как правило, первый и последний маршрутизаторы остаются неизменными, хотя даже это не обязательно Теоретически даже пакеты одного потока могут маршрутизироваться по-разному

На стороне клиента серверный ответ проходит по стеку сетевых протоколов, пока не достигнет клиентской программы В некоторых случаях это вызывает последующие запросы Например, Web-страница может содержать ссылки на графические файлы которые запрашиваются немедленно после получения основного документа. Если же TCP-трансфер окончен, то клиент должен завершить сеанс, т е сообщить серверу, что других запросов не будет. Для последующего взаимодействия с сервером придется устанавливать новое соединение

Брандмауэры и система NAT

Описанный процесс будет протекать по-другому, если на пути передачи данных присутствуют *брандмауэры* Это компьютеры (обычно их роль играют маршрутизаторы), которые предназначены для фильтрации проходящих данных. Брандмауэры блокируют запросы, поступающие от определенных компьютеров или портов или адресованные им Например, маршрутизатор серверной сети на рис 16 1 может быть сконфигурирован на передачу данных, адресованных порту 80 Web-сервера, поскольку это традиционный порт Web-трафика, в то же время маршрутизатор будет блокировать запросы к другим портам, в частности к порту 23 (используется в сеансах Telnet) Точно так же маршрутизатор клиентской сети может запрещать доступ к серверным портам клиентской системы, а также запросы от определенных серверов Если, допустим, пользователям клиентской сети не разрешен Web-доступ, маршрутизатор будет блокировать запросы, адресуемые порту 80 внешнего компьютера или поступающие от такого порта. Это сделает непосредственный Web-доступ невозможным (пользователи могут обойти такое ограничение с помощью внешнего *прокси-сервера* который переадресует Web-запросы через другой порт).

Брандмауэры являются важным средством защиты систем. В сети обычно работает много серверов Если злоумышленник найдет ошибку в конфигурации одного из них, то сможет через него атаковать остальные компьютеры локальной сети. Брандмауэры используются и для ограничения доступа к серверам Например, если серверы дистанционной регистрации (описываются в главе 21) должны быть доступны только локальным пользователям необходимо, чтобы брандмауэр блокировал внешний доступ к ним. Ограничения доступа должны задаваться и на самих серверах Подобная избыточность важна с точки зрения безопасности. преодолев одну преграду, злоумышленник тут же наткнется на следующую.

Некоторые маршрутизаторы реализуют систему NAT (Network Address Translation — трансляция сетевых адресов), которая позволяет объединять несколько компьютеров под одним IP-адресом и осуществлять другие преобразования IP-адресов NAT популярна в домашних сетях, в которых несколько систем совместно используют одно Internet-соединение Она также находит применение в организациях, где много компьютеров, но ограниченное число IP-адресов

Маршрутизатор, реализующий систему NAT, модифицирует IP-адреса проходящих пакетов. В наиболее распространенной конфигурации он подставляет вместо IP адреса и порта отправителя собственный IP-адрес и произвольный номер порта. Когда приходит ответный пакет, маршрутизатор проверяет, какой системе он назначил указанный порт и направляет пакет ей

Система NAT позволяет не только решить проблему нехватки IP адресов, но и ограничить доступ Поскольку маршрутизатор назначает порт по запросу, злоумышленник, находящийся во внешней сети не может посылать пакеты произвольному ком

пьютеру внутренней сети Таким образом, компьютеры защищены от прямых внешних атак, при условии что сам маршрутизатор не азломан и не назначил постоянный порт какой-нибудь внутренней системе (это может произоити если в локальной сети функционирует Web- или FTP-сервер).

Более подробная информация о брандмауэрах приведена в главе 17

Способы запуска серверов

Один из важнейших аспектов конфигурирования сервера связан со способом его запуска Сервер можно вызывать вручную, через сценарий запуска или через промежуточный сервер, который контролирует обращение к портам целевого сервера У каждого из этих способов есть свои преимущества и недостатки

Запуск вручную

Если сервер конфигурируется впервые, попробуйте запустить его вручную Для этого введите в командной строке его имя со всеми необходимыми аргументами. Вот как можно запустить например сервер `smbd` (является частью пакета Samba описываемого в главе 18)

```
# smbd
```

Многие серверы, вызванные подобным образом автоматически переходят в фоновый режим. Другие серверы остаются работать в интерактивном режиме, отображая сообщения об ошибках в целях отладки. Если необходимо, чтобы сервер работал длительное время без вмешательства пользователя, добавьте в строку вызова оператор &, который переведет сервер в фоновый режим.

Если на компьютере постоянно работает десяток серверов, запускать их вручную непрактично, особенно если система время от времени перезагружается. Более того, в случае внезапной перезагрузки (например, из-за сбоя питания) серверы на длительное время могут оказаться недоступными По этой причине предусмотрены автоматические способы запуска

Сценарии запуска

В главе 6 рассматривались основные сценарии запуска FreeBSD. Эти сценарии определяют перечень программ, вызываемых в процессе загрузки системы. Таким образом, чтобы заставить сервер автоматически запускаться нужно добавить ссылку на него в один из таких сценариев. Существуют две главные группы сценариев запуска

- **`/etc/rc.*`** Эти сценарии управляют стандартным процессом загрузки системы Сценарий `/etc/rc.network` особенно важен, поскольку содержит команды запуска многих сетевых серверов. Файл `/etc/rc.conf` содержит конфигурационные параметры, разрешающие или запрещающие конкретные серверы. Если, например, добавить в него строку `sshd_enable="YES"`, будет активизироваться сервер SSH (описывается в главе 21)
- **`/usr/local/etc/rc.d`** Сценарии этого каталога управляют запуском серверов, инсталлированных в каталоге `/usr/local`. Как правило, у каждого сервера свой сценарий. Многие пакеты включают в себя сценарии, имена которых оканчиваются на `.sh.sample` Если нужно разрешить запуск сервера, переименуйте сценарий, чтобы он оканчивался на `.sh`. Можно запускать и останавливать сервер вручную, вызывая соответствующий сценарий с аргументом `start` или `stop`

Какие бы сценарии ни использовались, все они организуют непрерывную работу серверов. Это означает, что серверы постоянно потребляют системные ресурсы, в основном память, но возможно, и немного процессорного времени. Серверы подключаются к выбранным портам и готовы в любой момент обработать поступивший запрос. Сервер может хранить в памяти информацию, способствующую повышению производительности, например кэш-буфер доменных имен, полученных от DNS сервера. Кроме того, сервер может периодически выполнять запланированные действия, например отправлять или загружать почту. Такая конфигурация удобна, если сервер долго загружается, хранит информацию о состоянии или выполняет действия по расписанию. Менее ресурсоемкая конфигурация, рассматриваемая ниже, предусматривает запуск серверов по запросу.

Использование суперсервера

Суперсервер — это особый тип сервера. Вместо того чтобы самостоятельно обрабатывать запросы, он подключается к портам других серверов, запуская их, когда поступают соответствующие запросы. Это более сложная конфигурация, чем в предыдущем случае, но у нее есть очевидные преимущества. Одно из них заключается в том, что суперсервер занимает гораздо меньше памяти, чем контролируемые им серверы. Это весьма немаловажно, если большинство серверов запускается лишь изредка. Серверами, непрерывно обрабатывающими запросы, не имеет смысла управлять подобным образом. Другое преимущество состоит в том, что суперсервер выполняет единообразный предварительный контроль доступа. Это очень удобно, если подчиненный сервер оснащен лишь минимальными средствами контроля. Важной особенностью суперсервера является то, что он может запускать другие серверы от имени непривилегированного пользователя.

Во FreeBSD в качестве суперсервера используется демон `inetd`. Как и многие другие серверы, он вызывается из стандартных сценариев запуска. Если он разрешен в файле `/etc/rc.conf` будет присутствовать следующая строка:

```
inetd_enable="YES"
```

При использовании программы `sysinstall` эту строку можно задать, включив опцию `inetd` в меню `Network Services Menu`, которое доступно из меню `FreeBSD Configuration Menu`. Чтобы узнать, запущен ли демон в настоящий момент, выполните команду **`ps ax | grep inetd`**. Если демон отсутствует, введите **`inetd -ww`** (опция `-ww` разрешает обращение к пакету TCP Wrappers, который используется демоном для обеспечения безопасности контролируемых серверов; см. главу 29).

Конфигурационный файл демона `inetd` называется `/etc/inetd.conf`. Он состоит из записей следующего вида:

```
служба тип протокол ожидание пользователь сервер опции
```

Строки, начинающиеся с символа #, являются комментариями. Рассмотрим назначение полей файла.

- *служба*. В первом поле содержится имя службы, определенное в файле `/etc/services`. Тем самым устанавливается привязка к номеру порта, который должен прослушиваться демоном `inetd` в ожидании запросов к серверу. Например, если строка начинается с имени `telnet`, демон будет контролировать порт 23, поскольку именно он закреплен за службой `telnet` в файле `/etc/services`.
- *тип*. Второе поле обычно содержит одно из двух ключевых слов: `stream` или `dgram`. Они определяют низкоуровневую обработку соединения с сервером.

- *протокол*. Третье поле обычно содержит ключевое слово `tcp` или `udp` задающее тип обрабатываемых пакетов. Стандартный файл `inetd.conf` содержит также примеры использования ключевого слова `tcp6`, которое обозначает TCP-службы протокола IPv6. Остальные значения встречаются редко.
- *ожидание* Четвертое поле содержит ключевое слово `wait` или `nowait`, определяющее способ запуска сервера: должен ли он отключиться сразу после обработки запроса или остаться в памяти в ожидании следующих запросов. Во втором случае демон `inetd` продолжит контролировать порт сервера и в случае поступления повторного запроса загрузит очередной экземпляр сервера.
- *пользователь* Пятое поле особенно важно, поскольку здесь задается пользователь, от имени которого демон `inetd` запускает сервер. Если серверу не нужны привилегии суперпользователя, то для повышения безопасности лучше указать непривилегированного пользователя, например `nobody` (стандартная учетная запись с минимальными привилегиями), или создать специальную учетную запись.
- *сервер* В шестом поле задается имя серверного файла. Обычно указывается полный путь к серверу, например `/usr/libexec/telnetd`. В этом поле может находиться ключевое слово `internal`, обозначающее внутренние службы самого демона `inetd`.
- *опции* В седьмом поле приводятся аргументы командной строки серверу. Чаще всего здесь указывается имя сервера, например `telnetd`.

Стандартный файл `inetd.conf` содержит записи для множества серверов, но почти все они превращены в комментарии. Чтобы разрешить тот или иной сервер, удалите символ комментарий и введите команду **`killall -HUP inetd`**, которая заставит демон повторно прочесть свой конфигурационный файл и учесть изменения. Некоторые серверы, упоминаемые в файле `inetd.conf`, вызываются из сценариев запуска. Соответствующие им записи должны быть превращены в комментарии.

В качестве примера рассмотрим следующие записи:

```
telnet  stream  tcp  nowait  root  /usr/libexec/telnetd  telnetd
tftp    dgram   udp  wait    root  /usr/libexec/tftpd    tftpd -s /tftpboot
```

Первая строка — это типичная конфигурация сервера Telnet. Он запускается всякий раз, когда на TCP-порт 23 поступают данные. Во второй строке конфигурируется сервер TFTP (Trivial FTP), который используется для передачи загрузочных файлов системам, загружающимся с сетевого сервера (он имеет мало общего с более распространенным протоколом FTP). Как правило, менять имеющиеся опции не приходится за исключением имени пользователя. Но нужно помнить, что некоторые серверы корректно функционируют только в привилегированном режиме. Это особенно справедливо для серверов дистанционной регистрации, принимающих запросы от множества пользователей. Если инсталлируется новый сервер и необходимо организовать его запуск через демон `inetd`, выясните в документации к серверу, какие установки ему соответствуют.

Совет Стандартный файл `inetd.conf` определяет довольно безопасную конфигурацию, поскольку ненужные серверы просто не запускаются. Тем не менее просмотрите файл и убедитесь в том, что лишние серверы действительно отключены. Базовое функционирование FreeBSD не зависит от серверов, запускаемых демоном `inetd`, поэтому, обнаружив непонятный сервер, можете смело превращать его запись в комментарий, не опасаясь катастрофических последствий. Проблемы возникнут лишь в том случае, если с этим сервером взаимодействуют другие системы.

Сетевая диагностика: утилита netstat

Конфигурация сети не всегда оказывается правильной В главах 14 и 15 рассматривались простейшие средства выявления неполадок в сети В частности, команды ping и traceroute позволяют убедиться том, что сеть функционирует. Для диагностики протоколов более высокого уровня существуют более сложные утилиты Одна из них — netstat, сообщающая информацию о сетевых соединениях.

Утилита netstat отображает разные сведения в зависимости от переданных ей параметров Назначение некоторых аргументов зависит от других параметров Мы рассмотрим лишь наиболее распространенные случаи применения этой довольно гибкой утилиты

На заметку В главе 14 было показано, как с помощью утилиты netstat вывести содержимое таблицы маршрутизации

В базовом варианте (без каких либо аргументов) утилита выдает следующий замысловатый отчет.

```
# netstat
Active Internet connections
Proto Recv-Q Send-Q  Local Address          Foreign Address  (state)
tcp4       0      0  halrloprillalar telnet nessus.2467      ESTABLISHED
tcp4       0      0  halrlopril.alar 1024   speaker.7101      ESTABLISHED
udp4       0      0  halrloprillalar 1019   speaker nfsd
Active UNIX domain sockets
Address  Type   Recv-Q Send-Q Inode     Conn Refs Nextref Addr
c928d780 stream      0      0      0 c928d820    0       0 /tmp/ X11 unix/X0
c928d820 stream      0      0      ) c928d780    0       0
```

Первый блок данных, помеченный как Active Internet connections, описывает соединения, установленные между локальной системой и другими компьютерами В показанном примере первые два соединения — это TCP-каналы (tcp4 в колонке Proto) между локальным компьютером (halrloprillalar) и компьютерами nessus и speaker В третьем случае происходит взаимодействие по протоколу UDP с компьютером speaker. Для каждого соединения указан номер порта или название службы, если оно определено в файле /etc/services Таким образом если сервер не удается запустить, а журнальные сообщения говорят о том, что порт занят выполните команду netstat и узнайте, какая программа захватила порт.

Второй блок данных, помеченный как Active UNIX domain sockets, описывает соединения, установленные в рамках локальной системы. В основном они связаны с доступом к X-серверу. Здесь приведено содержимое внутренних сетевых структур данных, которые интересны лишь специалистам.

По умолчанию утилита netstat не отображает информацию о серверах, которые ожидают запросов, но в настоящее время ни с кем не взаимодействуют Чтобы вывести список таких процессов, воспользуйтесь опцией -a Они будут помечены как LISTENING в колонке (state) Нужно также учитывать, что утилита пытается преобразовывать доменные имена в IP-адреса Чтобы отключить этот режим задайте опцию -n

Получение информации об интерфейсах

Опция -i заставляет утилиту выдавать сведения обо всех сетевых интерфейсах Указав опцию -I *имя_интерфейса*, можно получить отчет о конкретном интерфейсе например

```
# netstat -I fxp0 -n
Name  Mtu Network     Address           Ipkts Ierrs   Opkts Oerrs  Coll
fxp0 1500 <Link#1>  00:03:47:b1:ee:b8 7239      0    1982     0     0
fxp0 1500 192.168.1 192.168.1.6        7195          1944
```

Для большинства сетевых интерфейсов выдаются две записи. Первая из них касается низкоуровневого аппаратного интерфейса, а вторая отображает параметры стека TCP/IP, связанные с низкоуровневым интерфейсом. В обоих случаях указывается число принятых пакетов (`Ipkts`), входных ошибок (`Ierrs`), отправленных пакетов (`Opkts`), выходных ошибок (`Oerrs`) и коллизий (`Coll`). Коллизия происходит, когда пакеты двух систем поступают в сетевой кабель одновременно. Это неизбежная особенность некоторых сетевых технологий, которая приводит к снижению производительности сети.

Сбор статистических данных

Команда `netstat -s` отображает информацию о функционировании сети (число принятых и отправленных пакетов, ошибок, запросов на подключение и т.д.) группируемую по протоколам. Ее удобно использовать для анализа работы сетевых протоколов.

Практическое применение утилиты

С помощью утилиты `netstat` можно получить огромное количество данных о сетевых соединениях. К сожалению, их не так-то просто интерпретировать. Для этого нужно хорошо разбираться в особенностях работы сетевых протоколов. Кроме того, необходимо представлять себе особенности трафика в локальной сети. Например, в некоторых сетях происходит большое число коллизий. Они ведут к снижению производительности, поскольку, когда возникает коллизия, обе системы делают паузу (ее длительность выбирается случайным образом, чтобы свести к минимуму вероятность повторной коллизии), после чего повторно передают данные. В некоторых сетевых технологиях коллизии невозможно устранить, более того, их число увеличивается с ростом интенсивности трафика. Особенно подвержены этой проблеме старые коаксиальные сети Ethernet, а также сети на основе витой пары, в которых используются концентраторы. Если в такой сети показатель числа коллизий составляет 1–2%, то беспокоиться не о чем. Однако для сетей с коммутаторами это слишком высокий показатель.

Если в сети используются концентраторы и производительность сети не высока, имеет смысл перейти на коммутаторы. Они резко уменьшают число коллизий в сети и разрешают дуплексный обмен данными, при котором полоса пропускания сети используется гораздо рациональнее.

Используйте утилиту `netstat`, чтобы выяснить, какие показатели характерны для локальной системы и сети. Это позволит быстро выявлять аномалии в случае возникновения проблем.

Резюме

Базовые процедуры сетевого конфигурирования, рассмотренные в главах 14 и 15, следует выполнять перед запуском сетевых серверов и клиентов. Для более сложного конфигурирования необходимо разбираться в тонкостях функционирования сети. В частности, нужно знать, какую роль играют серверы и клиенты и как выбираются номера портов. Другой важный вопрос: как запускать серверы? Во FreeBSD это можно делать вручную, с помощью сценариев запуска или косвенным образом через суперсервер. Утилита `netstat` позволяет собирать информацию о состоянии сетевых интерфейсов и сетевых соединений. Это очень удобное средство выявления неполадок в сети.

Полный справочник по

Глава 17

Брандмауэры

Сети — неотъемлемый компонент современной коммуникационной инфраструктуры. К сожалению работа в сети связана с риском, поскольку сети любого размера подвержены взлому. Способы взлома бывают самыми разными, а их последствия зачастую разрушительны. Вот почему так важно заботиться о безопасности компьютеров подключенных к сети Этой теме будет посвящена глава 29, здесь же мы поговорим о конкретном средстве сетевой защиты: *брандмауэре* Он предназначен для ограничения доступа к компьютеру или группе компьютеров со стороны других компьютеров

В первую очередь необходимо выяснить, какова природа атак которым должен противостоять брандмауэр Далее мы узнаем, какие существуют типы брандмауэров и какие программы во FreeBSD используются для реализации этих механизмов защиты В завершение главы приведена информация о том, как программным путем организовать брандмауэр конкретного типа, называемый фильтром пакетов. Такой брандмауэр может защитить отдельный компьютер или (если его реализовать на маршрутизаторе) всю сеть

Источники угрозы

По своей сути брандмауэр является средством контроля доступа в сеть. Он предназначен для блокирования доступа на основании имен компьютеров, IP-адресов и других критериев Изначально предполагается, что во внешних сетях есть пользователи с плохими намерениями, которые могут попытаться взломать серверы, работающие на локальных компьютерах. Следовательно, необходимо ограничивать доступ к серверам. Кроме того, пользователи локальной сети сами могут атаковать другие компьютеры, поэтому брандмауэры часто конфигурируют для блокирования таких атак. Но даже если никто из пользователей не занимается ничем противоправным, в сеть может проникнуть вирус или "червь", способный самостоятельно выполнять различные атаки

Брандмауэры, как правило, используются для блокирования доступа между сетями, но безопасность *внутри* сети не менее важна. Чтобы снизить вероятность локальных атак, следует правильно конфигурировать локальные серверы и применять средства внутрисетевой защиты например Kerberos (`http://web.mit edu/kerberos/www/`).

Попытки взлома извне

Когда говорят сетевая безопасность , большинство людей представляют себе керов, пытающихся получить доступ к внутренним компьютерам корпоративной сети Имеется множество примеров подобного мошенничества среди которых наиболее распространены следующие

- **Взлом серверов регистрации.** Серворы, обеспечивающие регистрацию в системе (описываются в главе 21), позволяют пользователям дистанционно запускать программы. Взломщики атакуют эти серверы чтобы получить несанкционированный доступ в систему
- **Использование ошибок в программном коде сервера** Взломщики следят за выявлением ошибок в программных кодах популярных серверов и пытаются использовать их чтобы заставить сервер выполнять несанкционированные программы

- **Злоупотребление серверами открытой ретрансляции почты.** Излюбленная мишень спамеров — серверы, осуществляющие открытую ретрансляцию поступающей почты. Они используются спамерами для того, чтобы "замести следы" своей деятельности. Подробнее об этом рассказывается в главе 19.
- **Атаки типа "отказ от обслуживания"** Эти атаки нацелены на вывод из строя сетевого соединения путем перегрузки сетевого канала

На заметку В прессе под термином "хакер" часто понимают компьютерных преступников, но у этого слова есть более старое и достойное значение "Человек имеющий страсть к исследованию или написанию компьютерных программ, как правило, вне систематической инженерной деятельности" Многие из тех, кто пишут программы для FreeBSD, считают себя хакерами, поэтому для обозначения злоумышленников я предпочитаю употреблять термин "взломщик"

Большое разнообразие сетевых атак не позволяет написать универсальную программу, которая надежно предотвращала бы все возможные атаки Брандмауэр тоже блокирует лишь некоторые типы атак Рассмотрим рис 17 1, на котором изображена схема типичного применения брандмауэра В этой конфигурации роль брандмауэра играет маршрутизатор, который разрешает или запрещает доступ из внешних сетей в локальную сеть на основании заданных критериев Например, брандмауэр может запрещать внешний доступ к внутренним серверам дистанционной регистрации, таким как SSH, за исключением нескольких пользующихся доверием внешних IP-адресов. Это снизит шансы взломщика проникнуть в систему, даже если в конфигурации или программном коде локального SSH-сервера есть ошибка

На заметку Важным свойством высоконадежных систем является избыточность. Многие серверы имеют собственные средства контроля доступа на основании IP-адресов. Но из за возможных ошибок в программном коде или конфигурации желательно возводить несколько защитных барьеров.

Рис 17 1 Брандмауэр избирательно разрешает или запрещает доступ в сеть

Помимо ограничения доступа к известным серверам брандмауэры часто конфигурируют таким образом, чтобы они блокировали доступ к серверам, которые не должны работать. Например, доступ к порту SMTP (номер 25) может быть запрещен всем, кроме главного почтового сервера. Даже если на каком-нибудь внутреннем компьютере запущен неправильно сконфигурированный почтовый сервер, спамер не сможет эксплуатировать его. Следует запрещать доступ к портам протоколов, применяемых сугубо в локальных сетях, в частности протоколов совместного использования файлов и принтеров (описаны в главах 18 и 9).

К сожалению, брандмауэры не являются панацеей. У взломщика может быть учетная запись в системе, которой предоставляется тот или иной вид локального доступа. Кроме того, некоторые серверы *должны* быть доступны широкому кругу внешних систем. Например, почтовый сервер домена должен принимать почту от любого компьютера в Internet. Если такой сервер неправильно сконфигурирован или содержит ошибку, он может быть взломан, даже если между ним и сетью Internet находится брандмауэр. В некоторых сетях функционирует несколько брандмауэров, обеспечивающих многоуровневую защиту: они контролируют доступ к системам, обрабатывающим внешние запросы, и изолируют эти системы от уязвимых внутренних компьютеров.

Внутренние враги

Многие воспринимают брандмауэры как средство защиты внутренних компьютеров от проникновения из Internet, но защита может — и *должна* — быть двусторонней. Представьте, что вы являетесь администратором брандмауэра, который защищает десяток компьютеров с таким же количеством пользователей. Если один из этих пользователей является взломщиком, он может задействовать средства локальной сети для атаки внешних систем. Даже такая, казалось бы, "невинная" утилита, как `telnet`, способна стать инструментом взлома. Стоит взломщику неумело замаскировать свои действия, как следы неизбежно приведут к брандмауэру, через который пакеты взломщика вышли в глобальную сеть. Результатом может стать потеря репутации и даже уголовное преследование.

Брандмауэр можно сконфигурировать на блокирование определенных видов исходящего трафика. Предположим, пользователям нужно разрешить обращаться к Web- и FTP-узлам. Тогда следует разрешить исходящие запросы к портам 80 (WWW) и 21 (FTP) любого компьютера. К числу разрешенных протоколов часто относятся также SMTP (порт 25) и DNS (порт 53). Весь остальной исходящий трафик должен быть запрещен, что позволит снизить риск использования локальной сети в незаконных целях.

Как и в случае входящих данных, ограничение исходящих запросов не обеспечивает абсолютно надежной защиты. Если гипотетический внутренний взломщик объектом своей атаки изберет разрешенный порт, брандмауэр не сможет этому противостоять. Более того, атаке и переконфигурированию может подвергнуться сам брандмауэр (это справедливо и для внешних атак). Тем не менее указанное ограничение является важным компонентом общей стратегии сетевой безопасности.

Несанкционированное программное обеспечение во внутренней сети

Когда в сети работает более одного пользователя, всегда существует вероятность того, что один из них окажется ненадежным. Однако количество пользователей может и не иметь значения, поскольку зачастую дело не в самих пользователях, а в *несанкционированном коде*, к которому относят вирусы, "черви" и троянские кони. Программы первых двух типов особенно свирепствуют в Windows-системах. Вирусы заражают программы и распространяются в пределах одной системы. "Черви" имеют свойство

распространяться по сети, часто в виде почтовых вложений. Троянские кони — это программы, которые декларируют одно, а делают совсем другое. Впрочем, разница между тремя типами несанкционированных программ бывает довольно зыбкой

Вирусы и "черви" более характерны для Windows-систем, однако это не мешает брандмауэру, реализованному на базе FreeBSD, быть надежным средством, ограничивающим их распространение. Брандмауэр может защищать компьютеры, работающие под управлением любой операционной системы. Для него не важно, из какой системы поступил пакет.

Брандмауэр ограничивает распространение несанкционированного кода так же как он препятствует действиям внутренних взломщиков Если, например, "червь" запрограммирован устанавливать SMTP-соединение с внешней системой, то блокирование исходящих SMTP-соединений для всех компьютеров, кроме почтового сервера станет надежной преградой вирусу. Но, повторяю, стопроцентной защиты не существует. "Червь", использующий локальный почтовый сервер для рассылки зараженных" сообщений, сможет избежать ловушки.

Одной из важнейших мер борьбы с несанкционированным кодом является обучение пользователей. Не запускайте просто так программы, полученные из ненадежных источников, и запретите почтовым клиентам автоматически выполнять вложенный код Антивирусные программы для Windows, такие как Norton AntiVirus (`www.symantec.com`) и McAfee VirusScan (`www.mcafee.com`), хорошо умеют выявлять и удалять несанкционированный код

Во FreeBSD практически не встречаются ни вирусы ни "черви", поэтому и средства их обнаружения очень редки Но нельзя сказать, что FreeBSD полностью лишена проблем, просто они проявляются по-другому. Для FreeBSD более характерен взлом системы, о чем будет рассказываться в главе 29.

Типы брандмауэров

Приведенные выше сведения относятся ко всем брандмауэрам В действительности существуют различные типы брандмауэров и устройств, выполняющих аналогичные функции. В этом разделе мы поговорим об их отличительных особенностях.

Фильтры пакетов

Наиболее распространенным типом брандмауэра, о настройке которого будет рассказываться в разделе "Создание сценария фильтрации", является *фильтр пакетов*. Такой брандмауэр, сконфигурированный, например, на блокирование входящих запросов к порту 139 (протоколы совместного доступа к файлам SMB/CIFS), находит пакеты, адресованные этому порту, и либо удаляет, либо отклоняет их.

Фильтры пакетов легко реализуются с помощью стандартных средств FreeBSD Они могут ограничивать трафик на основании сетевого интерфейса, IP-адреса отправителя или получателя, исходного или целевого порта и других критериев. Одна программа позволяет создать фильтр, защищающий сеть от атак на множество серверов.

Правила фильтрации пакетов можно задавать как на маршрутизаторе для защиты всей сети, так и на отдельных компьютерах Таким способом осуществляется защита от определенных локальных атак

Маршрутизаторы NAT

В небольших сетях часто используются маршрутизаторы NAT (Network Address Translation — трансляция сетевых адресов). Хотя они не являются брандмауэрами, однако поддерживают многие их функции и могут реализовывать полноценные правила фильтрации. Основная задача маршрутизатора NAT — автоматически преобразовывать IP-адреса. Это выглядит так: клиент, имеющий один IP-адрес, инициирует передачу данных, а маршрутизатор модифицирует клиентские пакеты, чтобы они казались исходящими от другого IP-адреса, закрепленного за маршрутизатором. Когда приходит ответный пакет, маршрутизатор выполняет обратное преобразование, направляя пакет действительному клиенту. Аналогичным образом маршрутизатор NAT может менять адрес получателя.

Зачем нужны подобные преобразования? Они могут понадобиться в различных ситуациях:

- **Быстрое переконфигурирование сети**. Если необходимо поменять адрес сервера, система NAT позволит правильно переадресовывать трафик адресату на временной (или постоянной) основе.
- **Распределение нагрузки**. Маршрутизатор NAT можно сконфигурировать так, чтобы он распределял трафик, направляемый по определенному адресу, между несколькими системами. Это один из наиболее простых и быстрых способов выравнивания нагрузки.
- **Рациональное использование IP-адресов**. Маршрутизатор NAT разрешает сетевым клиентам иметь частные IP-адреса (например 192.168.*x.x*) и в то же время быть представленными в Internet под одним открытым IP-адресом (закрепленным за маршрутизатором). Это позволяет домашним пользователям и небольшим компаниям, которым выделяется один IP-адрес, подключать к Internet собственные локальные сети. Что касается крупных компаний, то система NAT дает им возможность расширять внутренние сети, не запрашивая большое число новых IP-адресов.

Последнее применение системы NAT наиболее распространено. Даже старый компьютер семейства 486 может использоваться в качестве маршрутизатора NAT для частной сети с десятком компьютеров, позволяя им использовать единый внешний IP-адрес.

В качестве средства "маскирования" частной сети система NAT обеспечивает определенные функции защиты, свойственные брандмауэрам. Внешние системы "видят" только маршрутизатор NAT и потому не могут напрямую подключиться к серверам, работающим во внутренней сети, как если бы брандмауэр блокировал внешние запросы. Соединения, устанавливаемые внутренними системами, пропускаются, если только не задан дополнительный набор правил фильтрации, ограничивающих исходящие запросы.

NAT — безусловно, полезное средство, но не лишенное ограничений. В частности, не все протоколы рассчитаны на использование NAT. В некоторых протоколах устанавливаются согласованные входящие и исходящие соединения (системы телеконференций, например). Другие протоколы встраивают информацию об IP-адресе в пакеты данных. Иногда такие ограничения можно обойти за счет специальных правил, но это не всегда возможно. Скажем, в ряде протоколов защищенной передачи данных IP-адрес шифруется, а следовательно, недоступен для системы NAT. Это не вызывает проблем при использовании распространенных протоколов, применяемых системами электронной почты или Web-броузерами, но более редкие протоколы, особенно предназначенные для шифрования, необходимо проверять на совместимость с NAT-программами FreeBSD.

Прокси-серверы

Еще одним типом брандмауэра является *прокси-сервер*. Это компьютер, который принимает запросы от внутренних систем, заданным образом обрабатывает их и пересылает конечному получателю. Когда приходит ответ, прокси-сервер обрабатывает его и направляет клиенту. Рассмотрим механизм интерпретации Web-запросов. Когда клиент инициирует исходящее соединение, запрос в действительности посылается внутреннему прокси-серверу, который генерирует почти аналогичный запрос и посылает его внешнему Web-серверу. Последний возвращает ответ локальному прокси-серверу, а тот, в свою очередь, переадресует его локальному Web-броузеру.

Прокси-серверы не являются маршрутизаторами. Такой сервер можно скрыть за брандмауэром, чтобы разрешить исходящий трафик с использованием определенных протоколов только компьютеру, на котором функционирует прокси-сервер. Допускается даже использовать прокси-серверы, находящиеся за пределами локальной сети. Прокси-сервер обрабатывает данные на более высоком уровне, чем фильтр пакетов. Это позволяет ему не только блокировать трафик на основании IP-адреса, но и распознавать имена файлов, а также сканировать эти файлы на наличие вирусов.

Помимо контроля доступа прокси серверы используются для повышения производительности сети. Например, Web-страница будет загружаться быстрее, если прокси-сервер удалит из нее рекламные баннеры. Но самое главное, прокси-серверы могут кэшировать данные. Это означает, что в случае повторного запроса к тем же самым данным прокси-серверу не придется запрашивать их в Internet.

Прокси серверы не всегда считаются брандмауэрами, но они могут использоваться как часть общей стратегии фильтрации. Вернемся к рис. 17.1. Система, обозначенная как "Маршрутизатор/брандмауэр", не обязательно является маршрутизатором. Это может быть компьютер, на котором выполняются прокси-серверы для протоколов внешнего доступа. В то же время, как отмечалось выше, прокси-сервер можно запустить во внутренней системе, разрешив маршрутизатору пропускать трафик соответствующего протокола только от одной этой системы.

Прокси-серверы неудобны тем, что для каждого используемого протокола обычно необходим отдельный сервер. Например, прокси-сервер Web-служб не будет обрабатывать SSH-трафик. Кроме того, не для всех протоколов имеются прокси-серверы. Эти серверы потребляют больше системных ресурсов, чем фильтры пакетов, так как они выполняют более сложную обработку запросов. Часто клиентские системы приходится дополнительно конфигурировать, чтобы они взаимодействовали с прокси-серверами. Внутренний взломщик может использовать прокси-сервер, на котором не ведется журнал событий, для того чтобы замести свои следы. По этим причинам я не буду подробно описывать прокси-серверы, а лишь перечислю наиболее популярные утилиты данного класса.

Средства настройки брандмауэров во FreeBSD

Во FreeBSD доступен целый ряд утилит, позволяющих реализовывать брандмауэры. Ниже будут описаны наиболее популярные из них. Сразу оговоримся, что это далеко не полный перечень.

Средства фильтрации пакетов: ipfw и IP Filter

Фильтрацию пакетов выполняет ядро FreeBSD, поскольку обработка пакетов на таком низком уровне – это прерогатива ядра. Таким образом, утилиты фильтрации пакетов являются всего лишь интерфейсами: они сообщают ядру о том, что следует

делать с пакетами которые удовлетаоряют определенным критериям Наиболее рас пространены две утилиты.

- **ipfw** Это встроенная сиетемная утилита фильтрации пакетов. Информацию о ней можно найти в документации к FreeBSD в частности на man странице ipfw
- **IP Filter** Этот пакет не так тесно связан с FreeBSD как ipfw, но он входит в состав системы и может инсталлироваться по умолчанию (поищите программы ipf и ipnat в каталоге /sbin) Дополнительная информация о пакете доступ на по адресу http://cpombs.anu.edu.au/ipfilter/.

Программа ipfw интегрирована в систему, поэтому именно она будет использо ваться в разделе "Создание сценария фильтрации". Ее преимуществом является то что она позволяет легко добавлять и удалять правила фильтрации. Это существенно упрощает настройку рабочей конфигурации. В случае пакета IP Filter необходимо модифицировать основной сценарий, а затем перезапустить программу ipf, чтобы она обновила всю конфигурацию

Программы ipfw и IP Filter содержат средства настройки NAT В первом случае соответствующая утилита называется natd во втором — ipnat. Конфигурирование демона natd описывается на с 358.

Распространенные прокси-серверы

Существует множестао различных прокси-серверов Одни из них предназначены для обработки конкретного протокола, другие поддерживают несколько протоколов. Некоторые серверы поставляются с готовыми клиентами которые работают с ними в связке В остальных случаях серверы зависят от конфигурационных опций внешних клиентов, таких как Web-броузеры, или правил фильтрацин, переадресующих исходящие пакеты прокси-серверу. На рис. 17 2 представлено окно настроек броузера Mozilla, в котором можно задать адрес прокси-сервера. Ниже перечислены наиболее широко используемые программы данного класса.

- **Squid**. Это популярный мультипротокольный прокси сервер, обрабатывающий трафик протоколов HTTP, FTP и Gopher. Основное назначение программы — кэширование даиных позволяющее ускорить обработку запросов локальных пользователей, часто обращающихся к одним и тем же Web-страницам. Дополнительную информацию можно найти на Web-узле www.squid-cache.org.
- **Privoxy** Этот прокси-сервер (www privoxy org) предназначен для удаления рекламных баннеров из Web-страниц и блокирования файлов "cookie" (небольших блоков данных, которые Web сервер передает броузеру для контроля за действиями пользователей).
- **DNDR**. Это один из немногих прокси-серверов DNS (вообще говоря, во FreeBSD DNS-сервер можно сконфигурировать как прокси-сервер имен). Для клиентов такая программа выполняет функции обычного DNS-сервера, хотя на самом деле переадресует все запросы внешнему серверу имен. Дополнительная информация о программе приведена на Web-узле http://dnrd.nevalabs.org
- **jftpgw** Это прокси-сервер FTP. Он контролирует доступ пользователей из локальной сети к внешним FTP-узлам и внешний доступ к внутренним FTP серверам. Адрес Web-узла программы таков: http://www.mcknight.de/jftpgw
- **Любой SMTP-сервер** Об этом почему-то редко упоминают, но любой SMTP сервер можно считать прокси-сервером Такие программы способны принн мать почту адресованную другим системам обрабатывать ее и ретранслиро

вать. Подобно обычным фильтрам, почтовые серверы могут фильтровать почту на основании различных критериев, например правил отсеивания спама

Некоторые из прокси-серверов, например `sendmail` (популярный SMTP-сервер) и Squid, входят в состав FreeBSD. К сожалению их конфигурирование столь существенно различается, что описание заняло бы слишком много времени (программа `sendmail` описывается в главе 19, но не в контексте прокси-серверов). Необходимые подробности можно найти в документации к прокси-серверам

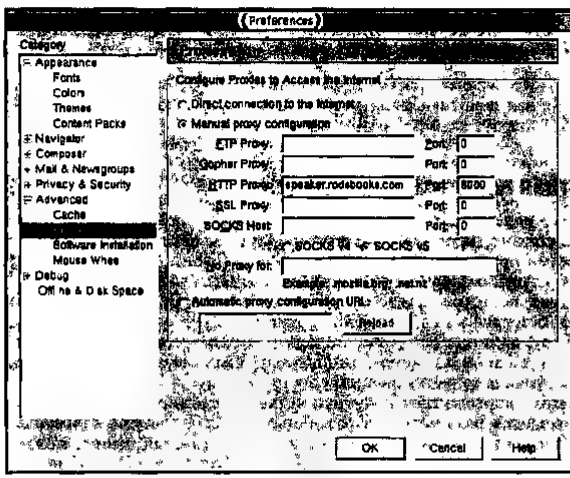


Рис 17 2 Клиентские Internet-программы позволяют задавать адреса прокси-серверов

Создание сценария фильтрации

Чтобы создать фильтр пакетов, необходимо запустить утилиту фильтрации, передав ей соответствующий конфигурационный файл, или написать сценарий который будет многократно вызывать утилиту, добавляя различные правила. В этом разделе мы рассмотрим, как организовать такой фильтр. В первую очередь будет показано, какие параметры ядра нужно включать и каковы общие принципы фильтрации. Далее будет рассмотрен процесс создания правил и их автоматической обработки

Необходимые параметры ядра

Прежде чем задавать правила фильтрации следует убедиться в том, что ядро поддерживает фильтрацию пакетов По умолчанию такая поддержка отсутствует, но стандартный сценарий фильтрации `/etc/rc.firewall` позволяет загрузить соответствующий модуль. Если осуществляется перекомпиляция ядра, добавьте один из нижеперечисленных параметров в конфигурационный файл (например, `/usr/src/sys/i386/GENERIC`)

```
options IPFIREWALL
options IPFIREWALL_VERBOSE
options IPFIREWALL_VERBOSE_LIMIT n
```

Эти параметры включают поддержку механизма фильтрации. Последние два указывают на необходимость журнальной регистрации проходящего трафика. Параметр *n* обозначает максимальное число пакетов, после которого регистрация будет прекращена. Это мера предосторожности на тот случай, если злоумышленник попытается атаковать систему, зная, что посылаемые им пакеты регистрируются, т. е. потребляют ресурсы процессора и занимают место на диске.

Если вы хотите поэкспериментировать с утилитой `ipfw`, не перекомпилируя ядро, загрузите необходимый модуль вручную с помощью команды **`kldload ipfw`**.

Не вводите показанную команду в дистанционном режиме. Стандартная политика фильтрации запрещает сетевой доступ, поэтому соединение тут же будет разорвано.

Стандартная политика фильтрации

В процессе переконфигурирования ядра задается *стандартная политика фильтрации*, т. е. специальное правило, применяемое ко всем пакетам, которые не были обработаны другими правилами. Оно является запрещающим, поэтому необходимо создавать явные правила, разрешающие определенные виды трафика.

Стандартная политика фильтрации является важной защитной мерой, так как обеспечивает безопасность всех портов, в том числе тех, о которых забыл администратор. К примеру, администратор мог не задать правило для порта 69 (редкий протокол TFTP), но если в одной из систем, защищаемых брандмауэром, работает сервер TFTP, стандартные правила запретят внешний доступ к нему.

Недостатком запрещающей политики являются сложности конфигурирования, поскольку приходится разбираться в различных моделях доступа, применяемых разрешенными клиентами и серверами. Упустив из виду важный нюанс, можно потом часами заниматься отладкой, выясняя, почему не работает та или иная программа. Кроме того, при добавлении нового сервера приходится переконфигурировать правила фильтрации.

Альтернативный подход заключается в создании разрешающей политики фильтрации. Сконфигурированный подобным образом компьютер пропускает любые пакеты за исключением тех, которые удовлетворяют явным правилам. Это проще реализовать, но все меры защиты должны быть явно предусмотрены администратором. Как правило, такая конфигурация является менее безопасной. Если все же нужно включить ее, введите правило, переопределяющее стандартную политику фильтрации ядра. Соответствующая команда выглядит так.

```
# ipfw add 65534 allow all from any to any
```

Если в ядре включен механизм фильтрации пакетов, но написание правил фильтрации пока не планируется, введите показанную команду, чтобы разрешить работу в сети. В противном случае FreeBSD будет блокировать любой сетевой трафик.

Имеется также специальный параметр компиляции ядра, задаваемый в файле конфигурации:

```
options IPFIREWALL_DEFAULT_TO_ACCEPT
```

Что необходимо разрешать, а что — запрещать

Большинство правил фильтрации связано с проверкой IP-адресов и номеров портов отправителя и получателя пакетов. Необязательно указывать всю эту информацию. Например, можно заставить систему пропускать пакеты, направляемые в порт 80 компьютера 172.21.25.101 с любого порта любого другого компьютера. Допускается указывать диапазоны портов и IP-адресов, а также задавать другие критерии фильтрации, такие как имя сетевого интерфейса.

Следует помнить о том, что существуют *привилегированные* и *непривилегированные* порты. Первые нумеруются от 1 до 1023. Во FreeBSD и других UNIX-системах эти порты может открывать только пользователь `root`. Именно с ними обычно работают стандартные серверы. Номера непривилегированных портов начинаются от 1024. Такие порты разрешено открывать программам, запускаемым рядовыми пользователями.

Не нужно создавать правила для всех возможных протоколов, поскольку существует стандартная политика фильтрации. Необходимо учитывать лишь те варианты, которые противоречат стандартной политике. Обычно разрешаются следующие виды трафика:

- **Трафик, направляемый санкционированным локальным серверам**. Если в сети работают серверы, нужно разрешить доступ к ним. Естественно, это не должны быть все желающие. Например, можно предоставить доступ к серверу SSH только компьютерам локальной сети.
- **Трафик, направляемый санкционированным внешним серверам**. Если пользователи имеют право обращаться к Web-серверам, FTP-серверам и т.д., нужно разрешить исходящий трафик к ним. Не забывайте и о таких распространенных протоколах, как DNS (UDP-порт 42) и DHCP (UDP-порт 67). В случае необходимости задайте явные IP-адреса внутренних или внешних систем.
- **Ответный трафик от санкционированных серверов**. Следует разрешать ответные пакеты как от внутренних, так и от внешних серверов. Предостерегать доступ лишь на основании номера порта рискованно, поскольку злоумышленник может провести атаку на такой порт. Более распространенный подход заключается в том, чтобы пропускать пакеты, передаваемые в рамках соединения с непривилегированным портом. Такая конфигурация позволяет клиентам устанавливать соединение (если другие правила разрешают это) и получать данные от внешнего сервера.

Определенные виды доступа, как правило, запрещены:

- **Трафик от несанкционированных узлов**. Некоторые серверы не должны быть доступны кому попало. Например, локальные файловые серверы и серверы печати предназначены лишь для пользователей локальной сети. Брандмауэр может ограничивать доступ к ним на основании комбинации номера порта и IP-адреса.
- **Доступ к неиспользуемым серверам**. Сервер, который не работает, не представляет угрозы, но бывает, что неиспользуемые серверы запускаются по недосмотру. Блокирование доступа к портам таких серверов позволяет избежать ненужных проблем.
- **Неправильные пакеты**. Пакеты с неправильным содержимым могут вызывать проблемы в сети или свидетельствовать о попытке взлома. Например, если обнаруживается пакет, который поступает из внешнего интерфейса и при этом имеет исходный IP-адрес, принадлежащий локальной сети, то почти наверняка имеет место подделка IP-адреса. Брандмауэры должны выявлять и блокировать подобные атаки.

Решение о том, какой трафик необходимо разрешать или запрещать зависит от разных факторов. какие серверы функционируют, где сконфигурирован брандмауэр (на маршрутизаторе, клиентском или серверном компьютере), чему отдается предпочтение — безопасности или удобству пользователей и т.д.

Создание правил

Зная, где функционирует брандмауэр и какие протоколы он должен распознавать можно приступать к настройке правил фильтрации. Правила задаются в виде упорядоченного списка. Если пакет соответствует критерию, указанному в правиле, он либо принимается, либо отвергается, в зависимости от директивы. Порядок правил чрезвычайно важен. Если, к примеру, одно из первых правил разрешает трафик, направляемый в TCP-порт 80 от любого IP-адреса, то последующее правило, запрещающее весь трафик от конкретного IP-адреса, не будет выполняться в случае, когда данные посылаются в порт 80. Кроме того, чем раньше встречается правило, тем меньше времени тратится на его обработку. Об этом следует помнить, создавая правила для обработки трафика, возвращаемого внешними серверами.

Синтаксис вызова утилиты ipfw

Правила фильтрации создаются с помощью утилиты `ipfw`. Синтаксис ее вызова чрезвычайно сложен и подробно описан на man странице. В упрощенном виде он выглядит так:

```
ipfw [-N] инструкция [индекс] [директива] [log] протокол [опции]
```

Рассмотрим назначение параметров.

- `-N` По умолчанию утилита `ipfw` работает с адресами узлов и номерами портов. Опция `-N` заставляет утилиту преобразовывать эту информацию в имена, используя DNS и файл `/etc/services`.
- *инструкция*. Это действие, выполняемое по отношению к правилу. Допустимые инструкции перечислены в следующем подразделе.
- *индекс*. Это число от 0 до 65535, обозначающее положение правила в списке. Если опустить данный параметр, система автоматически сгенерирует номер, который на 100 больше, чем номер самого последнего правила, за исключением правила по умолчанию (имеет номер 65535).
- *директива*. Это действие, выполняемое по отношению к пакету. Директива `allow` разрешает принимать пакет. Ее синонимами являются директивы `accept` и `pass`. Директивы `deny` и `reject` означают блокирование пакета, но разными способами. Директива `deny` заставляет систему игнорировать пакет, чтобы отправитель думал, будто пакет потерялся или компьютер-адресат выключен. Директива `reject` заставляет систему вернуть отправителю сообщение о том, что компьютер или порт недоступен (как если бы маршрутизатор не смог найти компьютер или на компьютере отсутствовала программа, контролирующая заданный порт). Директива `deny` сводит к минимуму риск определенных атак типа "отказ от обслуживания" и усложняет взломщику задачу сканирования системы. В то же время директива `reject` позволяет скрыть факт наличия брандмауэра. Обычно предпочитают указывать директиву `deny`, но есть важное исключение: порт 113. Он используется сервером `ident`, который пытается определить имя пользователя, запустившего клиентское приложение. Директива `deny` существенно замедляет его работу. Директива `count` заставляет систему увеличить счетчики правила, но не выполнять других действий. Обработка пакета будет продолжена последующими правилами.

- `log` Этот параметр заставляет утилиту `ipfw` выводить информацию о совпадениях на системную консоль.
- *`протокол`* Это протокол проверяемых пакетов. Наиболее распространенные значения — `tcp`, `udp` и `icmp`. Ключевому слову `ip` или `all` соответствуют пакеты любого протокола.
- *`адреса`* Спецификация адреса сама по себе довольно сложна, поэтому описывается отдельно (с. 355). В этой спецификации могут присутствовать IP-адреса и номера портов отправителя и получателя, а также имя или адрес сетевого интерфейса.
- *`опции`*. Утилита `ipfw` поддерживает различные опции, задающие тип пакета. Например, ключевому слову `established` соответствуют пакеты, являющиеся частью существующего соединения, ключевому слову `setup` — пакеты, посылаемые при попытке установить соединение, ключевому слову `in` — входящие пакеты, а ключевому слову `out` — исходящие. Особенно полезна опция `established`, которая позволяет создавать правила, разрешающие ответный трафик.

Инструкции утилиты ipfw

Список правил обычно представляется в виде сценария, выполняемого на этапе начальной загрузки системы. Большинство правил добавляется с помощью инструкции `add`. Остальные инструкции чаще используются в интерактивном режиме.

- **`add`** — добавляет правило в список.
- **`delete`** — удаляет правило из списка.
- **`list`** — выводит весь список правил или только заданное правило.
- **`flush`** — удаляет все правила из списка, за исключением правила по умолчанию.
- **`resetlog`** — сбрасывает счетчик совпадений для правила. Можно создать задание демона `cron`, которое будет периодически сбрасывать счетчики регистрируемых правил, чтобы после получения максимально допустимого числа пакетов совпадения продолжали регистрироваться.

Синтаксис адресов утилиты ipfw

Спецификация адреса имеет следующий вид:

```
from адрес/маска [порт] to адрес/маска [порт] [via интерфейс]
```

Каждый из ее компонентов заслуживает отдельного пояснения.

- Пакет TCP/IP содержит адреса отправителя и получателя. Они должны задаваться с помощью ключевых слов `from` и `to`. Если адрес не важен, укажите ключевое слово `any`.
- Адрес может быть задан в традиционном виде (например, `172.27.145.31`) или в виде сетевого адреса с маской формата CIDR (`172.27.145.0/24`) либо маской четырехбайтового формата (`172.27.145.0.255.255.255.0` — обратите внимание на двоеточие, отделяющее адрес от маски). Вместо адреса разрешается указывать доменное имя, но это менее безопасный способ, поскольку он зависит от DNS-сервера, который может быть взломан. Ключевому слову `any` соответствует любой адрес.
- Аргумент *`порт`* — это номер порта для протоколов, которые поддерживают это понятие (TCP и UDP). Разрешается опускать номер порта, указывать несколь-

ко портов через запятую (например `25 80`) или задавать диапазон номеров через дефис (`1 1024`)

- Если необходимо, чтобы правило применялось к трафику конкретного сетевого интерфейса, воспользуйтесь ключевым словом `via`. Аргумент *интерфейс* представляет собой IP адрес или имя интерфейса (например `fxp0`)

Примеры правил

Теперь рассмотрим примеры использования команды `ipfw`, чтобы разобраться в ее синтаксисе. Сначала проанализируем команду уже приводившуюся на с 352:

```
# ipfw add 65534 allow all from any to any
```

Эта команда добавляет правило с номером 65534 которыи на единицу меньше чем номер правила по умолчанию (65535) Новая команда заставляет систему принимать пакеты от любого компьютера и пропускать пакеты адресованные любому компьютеру В сущности оно переопределяет стандартную политику фильтрации. Если теперь просмотреть список правил, в нем будет два элемента

```
# ipfw list
65534 allow ip from any to any
65535 deny ip from any to any
```

Новое правило идентично стандартному, только вместо директивы `deny` присутствует директива `allow` (при отображении правила утилита `ipfw` преобразовала ключевые слова `all` в `ip`; это синонимы, так что замена несущественна).

Теперь предположим, что требуется создать настоящий брандмауэр на клиентском компьютере, где работает почтовый сервер SMTP. Чтобы восстановить стандартную политику фильтрации выполните одну из следующих команд

```
# ipfw delete 65534
# ipfw flush
```

Первая из них удаляет одно, ранее добавленное, правило, а вторая удаляет все правила, кроме стандартного В нашем случае обе команды эквивалентны. С этого момента система перестанет пропускать входящий и исходящий трафик Вот как разрешить входящий трафик SMTP-сервера.

```
# ipfw add allow tcp from any to 172.23.45.67 25
```

Здесь предполагается, что сервер имеет IP-адрес 172 23 45 67 SMTP-серверы обычно принимают почту от любого компьютера, поэтому после ключевого слова `from` указано `any` Теперь порт 25 открыт для входящего трафика, но не для исходящего Следующая команда разрешает серверу отправлять данные:

```
# ipfw add allow tcp from 172 23.45.67 25 to any
```

Чтобы проверить работу сервера, отправьте ему почтовое сообщение из другой системы или воспользуйтесь программой Telnet другого компьютера для доступа к порту 25. Но сам сервер не сможет отправлять почту, поскольку брандмауэр блокирует DNS-запросы, отправляемые через UDP порт 53 Показанные ниже команды решают эту проблему:

```
# ipfw add allow udp from 172 23.45.1 53 to 172.23.45 67
# ipfw add allow udp from 172.23.45.67 to 172 23 45 1 53
```

Они разрешают системе взаимодействовать через UDP-порт 53 с компьютером 172 23 45 1 (адрес DNS-сервера в этом примере). Если есть несколько DNS-серверов, можно для каждого из них задать отдельные правила

Приведенные выше правила касаются работы серверов, а как насчет клиентов? В первую очередь необходимо разрешить определенные виды исходящих соединений например HTTP и Telnet.

```
# ipfw add allow tcp from 172 23 45 67 1025 65535 to any 23 80
```

Эта команда говорит о том, что с любого непривилегированного порта можно ус танавливать соединения с портами Telnet (25) и HTTP (80) любого компьютера Но как организовать передачу обратного трафика? Для этого нужно разрешить *установленные* соединения с любым непривилегированным портом.

```
# ipfw add allow tcp from any 23,80 to 172 23 45.67 1025 65535↵
  established
```

Поскольку соединение устанавливал клиент, все возвращаемые пакеты соответствуют параметру `established` А вот попытка установить соединение с сервером, который работает на непривилегированном порту, потерпит неудачу

Если нужно разрешить любые исходящие соединения, не указывайте номера портов в двух предыдущих командах Это откроет доступ ко всем сетевым протоколам, но в то же время упростит задачу взломщику которому станет проще генерировать подельные пакеты

Показанные выше правила служат лишь демонстрационным целям Реальные брандмауэры содержат гораздо больше правил Приведенные правила характерны для клиентских и серверных компьютеров В случае маршрутизатора правила будут схожими, только наверняка добавится спецификация интерфейса для защиты от подельных адресов

Совет В файле `/etc/rc.firewall` содержится большое число правил. Изучите этот сценарий, чтобы лучше узнать, как создаются различные правила

Автоматизированная обработка сценария фильтрации

Разумеется, никому не хочется вводить правила фильтрации вручную при каждом запуске системы Чтобы избежать этого, поместите все правила в сценарий. Такой сценарии подключается в файле `/etc/rc.conf`

```
firewall_enable="YES"
firewall_type="client"
```

Эти параметры заставляют систему выполнять сценарии `/etc/rc.firewall` на этапе начальной загрузки Тип брандмауэра обозначен как `client` Возможные типы таковы

- **`open`** - разрешен доступ к любой системе
- **`closed`** — запрещен любой доступ;
- **`client`** разумная конфигурация для многих клиентских систем
- **`simple`** — начальная конфигурация для маршрутизатора,
- **`unknown`** - не загружаются никакие правила, кроме стандартного

С помощью параметра `firewall_type` можно задать другой файл сценария на пример.

```
firewall_type="/usr/local/etc/firewall"
```

Такой файл должен содержать команды вызова утилиты `ipfw` только имя самой утилиты указывать не нужно

```
add allow udp from 172 23 45 1 53 to 172.23.45 67
add allow udp from 172 23.45.67 to 172 23.45 1 53
```

Сценарии `/etc/rc.firewall` загружает модуль фильтрации пакетов, а также за данные правила Это удобно, поскольку не требуется перекомпилировать ядро По экспериментируйте с различными параметрами, чтобы понять, к каким результатам они приводят

Совет Введите `/etc/netstart`, чтобы перезапустить сетевую подсистему включая сценарии фильтрации пакетов, без перезагрузки компьютера Если IP-адреса назначаются по протоколу DHCP, введите `killall dhclient; /etc/netstart`, иначе система потеряет свой IP-адрес.

Конфигурирование NAT

Чтобы FreeBSD-система начала функционировать как маршрутизатор NAT акти визируйте демон `natd`. На соответствующей man-странице описан довольно сложный синтаксис вызова демона Но большинство опций касается действительно сложных аспектов конфигурирования NAT, таких как распределеиие нагрузки В простой конфигурации когда маршрутизатор "скрывает" локальную сеть под внешним IP-адресом, достаточно поместить следующие строки в файл `/etc/rc.conf`:

```
gateway_enable="YES"
firewall_enable="YES"
firewall_type="simple"
natd_enable="YES"
natd_interface="fxp0"
natd_flags=""
```

Укажем ключевые особеииоети такой конфигурации.

- Обычно маршрутизатор NAT имеет два сетевых интерфейса, одии из которых подключен к локальной сети, а другой — к выделенной линии. Параметр `gateway_enable` разрешает маршрутизацию между этими интерфейсами
- Маршрутизатор NAT может содержать правила фильтрации пакетов как и любой другой маршрутизатор Система NAT позволяет не беспокоиться о непосредственных атаках на локальную сеть, однако возможны атаки ил сам маршрутизатор, а также атаки берущие начало в локальной сети.
- Параметр `natd interface` задает сетевой интерфейс, подключенный к Internet Это может быть выделеннав линия или обычный коммутируемый PPP-канал Если по ошибке указать локальный интерфейс, система начнет функционировать наоборот "скрывать" всю сеть Internet под одним IP-адресом
- Если необходимо запустить сервер, контролируемый маршрутизатором NAT воспользуйтесь параметром `natd flags`. В частности флаг `redirect_port` включает переадресацию запросов Например строкой

```
natd flags="-redirect_port tcp 192.168 1.3:80 80"
```

задается перенаправление пакетов, проходящих через внешний TCP порт 80 маршрутизатора, порту 80 компьютера 192.168.1.3

Как и в случае фильтрации пакетов, система NAT требует поддержки со стороны ядра Параметр `natd_enable` в файле `/etc/rc.conf` заставляет систему загрузить соответствующий модуль. Если осуществляется перекомпиляция ядра добавьте в конфигурационный файл следующие строки

```
options IPFIREWALL
options IPDIVERT
```

Резюме

Брандмауэры — чрезвычайно важное средство защиты сетей. Любая система, постоянно подключенная к Internet, должна защищаться брандмауэром, иначе ей не устоять перед сетевыми атаками.

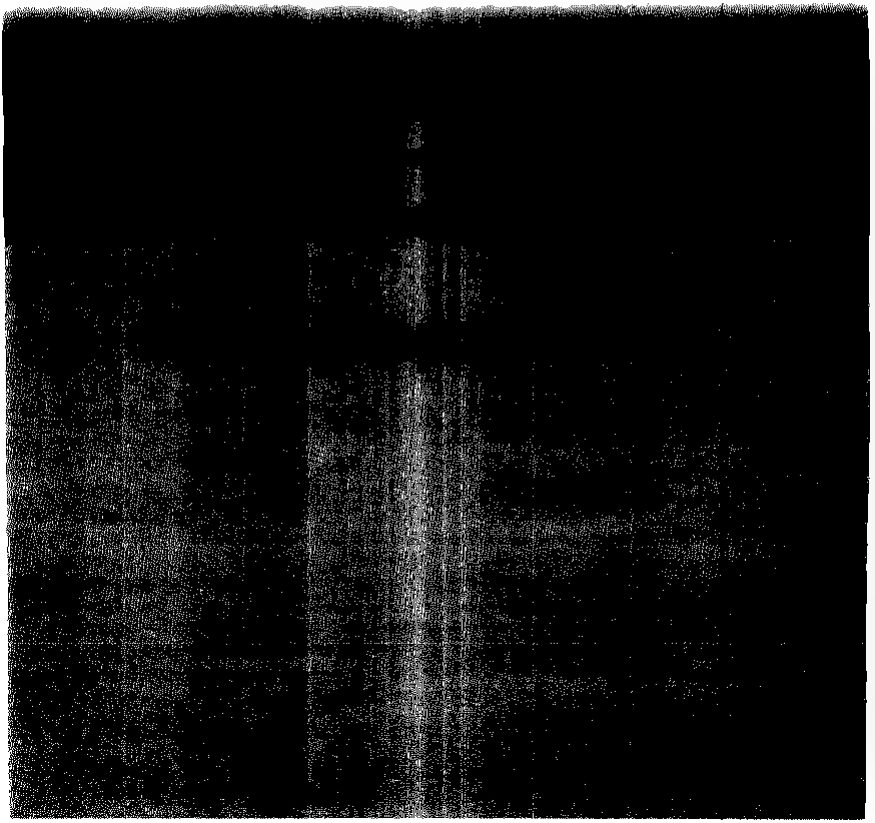
Брандмауэры бывают двух основных видов: фильтры пакетов и прокси-серверы. Первые реализуются во FreeBSD с помощью программы `ipfw` или IP Filter. Прокси-серверы реализуются с помощью множества утилит, различающихся типом предоставляемой защиты. Если используется утилита `ipfw`, необходимо задать набор правил, оговаривающих порядок доступа к портам системы. Эти правила должны пропускать лишь непосредственно разрешенный трафик, запрещая все остальные пакеты, что позволит защитить нерегулярно запускаемые серверы.

Полный справочник по FreeBSD

Часть IV

Серверы

Полный справочник по



Во многих сетевых операциях происходит передача файлов. В виде файлов представляются почтовые сообщения, Web-страницы, сообщения Usenet и т.д. Таким образом, серверы соответствующих протоколов обмениваются файлами, хотя это не всегда очевидно. Но есть серверы, задачи которых напрямую связаны с управлением файлами. Они позволяют пользователям копировать файлы на сервер и загружать файлы с сервера. Эта глава посвящена конфигурированию и применению трех распространенных файловых серверов:

- FTP (File Transfer Protocol — протокол передачи файлов) — давнишний стандартный протокол сети Internet.
- NFS (Network File System — сетевая файловая система) — популярный протокол совместного доступа к файлам в UNIX.
- Samba — позволяет клиентам Windows получать совместный доступ к UNIX-системам по протоколам SMB (Server Message Block — блок серверных сообщений) и CIFS (Common Internet File System — единая файловая система для Internet).

Но для начала поговорим о том, какие вообще существуют файловые серверы и чем они отличаются друг от друга.

Типы файловых серверов

Файловые серверы существенно различаются по своим функциональным возможностям. В основном это связано с различием реализуемых ими протоколов. Ниже будут описаны серверы двух типов: серверы доставки файлов и серверы совместного доступа к файлам, а также сделан краткий обзор наиболее популярных файловых серверов.

Серверы доставки файлов и серверы совместного доступа к файлам

Под термином *сервер доставки файлов* понимается наиболее общий тип файлового сервера. Такие серверы позволяют пользователям запускать клиентские программы с привязкой к файловой системе сервера. Пользователь может выбирать файлы как на стороне клиента, так и на стороне сервера, пересылая файлы с одного компьютера на другой. Самым популярным протоколом передачи файлов является FTP.

Слабым местом серверов доставки файлов являются ограниченные возможности непосредственного управления файлами удаленной системы. Пользователю не всегда нужна локальная копия файла. Предположим, он хочет изменить файл или напрямую загрузить его в локальное приложение. Серверы доставки файлов затрудняют внесение таких динамических изменений. В этом случае на помощь приходят *серверы совместного доступа к файлам*. Предполагается, что клиентский компьютер делает серверные файлы доступными локальным приложениям, как если бы серверная файловая система представляла собой локальный раздел жесткого диска. Программы, запущенные на клиентском компьютере, могут непосредственно загружать файлы с сервера и редактировать их. Наиболее популярные серверы этого типа — NFS и Samba.

Серверы совместного доступа в сущности реализуют те же функции, что и серверы доставки файлов, но с дополнительными расширениями. Как правило, они лучше интегрируются с файловой системой клиента. В частности, NFS поддерживает UNIX-атрибуты файлов, а в протоколах SMB/CIFS поддерживаются DOS-атрибуты "только чтение", "скрытый" и "архивный". В остальном серверы обоих типов функционируют

сходным образом, позволяя копировать файлы в обоих направлениях. Более того, есть клиенты, интерпретирующие FTP как протокол совместного доступа к файлам, а в пакет Samba входит утилита `smbclient` реализующая FTP подобный интерфейс к серверам Samba.

Дополнительные возможности файловых серверов

Помимо базовых функций передачи файлов рассматриваемые серверы поддерживают ряд дополнительных возможностей.

- **Интерпретация имен файлов.** В большинстве серверов ограничения на длину имен файлов являются довольно либеральными. Часто детали обработки имен (например, чувствительны ли они к регистру символов) определяются операционной системой.
- **Передача метаданных.** В некоторых протоколах вместе с файлами передаются и метаданные (данные, описывающие файл), в частности информация о владельцах и правах доступа. Эти метаданные важны для серверов совместного доступа, так как помогают серверу обеспечивать совместимость с файловой системой клиента.
- **Листинги файлов.** При работе с файловой системой сервера полезно видеть список доступных файлов. Все три описываемых в настоящей главе сервера позволяют получать такие листинги, но некоторые более примитивные файловые серверы не умеют этого делать. В таком случае нужно знать имена файлов, участвующих в запросе.

Важную роль могут играть и возможности клиентов. Для протокола FTP, например, существует множество клиентов — от исходной текстовой утилиты `ftp` до сложных графических приложений. Серверы совместного доступа обычно вызываются из встроенных утилит клиентских операционных систем, которые делают файловую систему сервера доступной в виде локального дискового раздела. О сетевых клиентах FreeBSD будет рассказываться в главе 24.

Распространенные файловые серверы и их применение

Приведенный ниже перечень поможет читателям сориентироваться в многообразии файловых серверов.

- **Серверы FTP.** Как уже отмечалось, FTP — один из старейших протоколов передачи файлов, но до сих пор очень популярный. Он реализован в таких серверах, как WU-FTP, ProFTPd и BSD FTP. В следующем разделе будет описано функционирование сервера BSD FTP во FreeBSD. Если пользователи имеют учетные записи на сервере FTP, то поддерживается двусторонний обмен файлами. Существует также особый *анонимный режим*, в котором всем пользователям назначается общая "анонимная" учетная запись (из соображений безопасности в анонимном режиме обычно разрешается лишь чтение файлов с сервера). Многие пользователи сегодня не подозревают об использовании протокола FTP, однако он продолжает играть важную роль, поскольку Web-страницы часто содержат ссылки на бинарные файлы, хранящиеся на FTP-серверах. Существенным недостатком протокола является отсутствие шифрования, вследствие чего ни пароли, ни передаваемые данные не защищены в достаточной степени.
- **Серверы NFS.** Протокол NFS разработан компанией Sun Microsystems и предназначается для организации совместного доступа к файлам в среде UNIX. Он поддерживает все традиционные функциональные возможности файловой системы UNIX и потому применяется достаточно широко. Безопасность протоко-

ла основана на модели *надежных узлов*, в которой аутентификацию пользовате лей и контроль доступа к файлам выполняют клиенты. Эта модель была разумной в былые времена, когда UNIX-компьютеры были малочисленны и надежно контролировались. В современных условиях она слишком рискованна Вот почему так важно тщательно проверять права доступа к серверам NFS.

- **Серверы SMB/CIFS** Протоколы SMB/CIFS используются в среде Windows для организации совместного доступа к файлам и принтерам Они поддерживают функции файловых систем DOS и Windows, но не UNIX. Пакет Samba реализует серверную часть этих протоколов в UNIX Во многих отношениях сервер Samba является более гибким, чем сервер SMB/CIFS, встроенный в Windows Это позволяет интегрировать компьютеры, работающие под управлением FreeBSD и Windows, в единую сеть В протоколах SMB/CIFS контроль доступа осуществляется на основании имен пользователей и паролей Пароль может шифроваться, но остальные данные нет
- **Серверы AppleTalk** Протокол совместного доступа к файлам AppleTalk был раз работан компанией Apple для компьютеров Macintosh FreeBSD-систему можно превратить в сервер AppleTalk с помощью пакета Netatalk (`http://netatalk.sourceforge.net`). Он ориентирован на версии Mac OS, более ранние, чем Mac OS X поэтому мы не будем его рассматривать Операционная система Mac OS X реализована на базе ядра UNIX (включая немало заимствований из FreeBSD) и поддерживает совместный доступ к файлам по протоколам NFS, AppleTalk и SMB/CIFS Как и в предыдущем случае, аутентификация основана на проверке имени пользователя и пароля, а шифрование пароля зависит от установок клиента и сервера
- **Серверы `rcp`** Утилита `rcp` простейшее средство передачи файлов. Она напоминает обычную команду `cp` для локального копирования файлов, но поддерживает сетевые операции Подобно NFS, утилита `rcp` основана на модели надежных узлов, а параметры доступа хранятся в пользовательских файлах `.rhosts` Эта утилита позволяет легко и быстро копировать файлы по сети, но в настоящее время она считается слишком опасной
- **Серверы SSH** Пакет SSH (Secure Shell) является сервером дистанционной ре гистрации и потому описан в главе 21 Но в нем есть и средства копирования файлов Это, в частности, клиентская утилита `scp`, напоминающая `rcp` В последних версиях пакета появились также утилиты передачи файлов, имеющие свойства FTP-клиентов В частности утилиты `sftp` и gFTP имеют FTP-подобный интерфейс но реализуют зашифрованную передачу данных через сервер SSH (утилита gFTP изначально являлась клиентом FTP и до сих пор может функционировать подобным образом)
- **Web-серверы** Web-серверы работают по протоколу HTTP (Hypertext Transfer Protocol) и в основном осуществляют одностороннюю доставку файлов с сервера на клиент, хотя возможна передача файлов и в обратном направлении Аутентификация обычно не применяется Файлы, переданные Web-сервером немедленно отображаются клиентской программой (Web-броузером). В главе 20 описано конфигурирование наиболее популярного Web-сервера Apache

FTP-серверы используются для межплатформенной передачи файлов, а также в том случае, когда клиентам не нужно динамически менять данные, хранящиеся на сервере Аутентификация возможна (она позволяет отдельным пользователям получать доступ к файлам на сервере), но на практике нежелательна, поскольку пароли передаются в незашифрованном виде Для этих целей следует использовать пакет SSH

Серверы совместного доступа к файлам применяются вместе с клиентами соответствующих операционных систем (UNIX для NFS, DOS, Windows или OS/2 для Samba Mac OS для Netatalk). Исключения, например подключение клиентов Windows через NFS встречаются очень редко. К ним прибегают когда в сети доминируют системы определенного типа, а клиентов другого типа всего один два. В этом случае проще инсталлировать на клиентском компьютере средства поддержки стороннего протокола. Иногда применение того или иного протокола диктуется соображениями безопасности

FTP-сервер

Базовая конфигурация FTP-сервера во FreeBSD очень проста, если используется стандартный сервер BSD FTP. Для запуска такого сервера необходимо внести изменения в настройки демона `inetd`. Более сложная конфигурация потребуется в случае анонимного FTP-сервера

Распространенные FTP-серверы

Протокол FTP появился достаточно давно, поэтому для него существует целый ряд серверов. Ниже перечислены серверы, доступные во FreeBSD

- **BSD FTP**. Стандартный FTP-сервер FreeBSD впервые появился в 4.2 BSD. Он довольно просто конфигурируется и инсталлируется по умолчанию в большинстве систем
- **WU-FTPD**. Демон WU FTPD (`www.wu-ftpd.org`) является популярным FTP сервером для многих платформ, но во FreeBSD он используется редко. В нем было выявлено немало ошибок, связанных с безопасностью
- **ProFTPd** — довольно гибкий и легко расширяемый FTP сервер (`http://proftpd.linux.co.uk`). Синтаксис его конфигурационного файла аналогичен синтаксису, который используется сервером Apache
- **oftpd** — небольшой FTP-сервер, обеспечивающий безопасность анонимного FTP-узла. Дополнительную информацию о нем можно найти по адресу `http://www.time-travellers.org/oftpd`.
- **twoftpd** — еще один FTP-сервер, достоинство которого — простота и безопасность. В отличие от других серверов, он не может выполнять внешние команды. После того как пользователь регистрируется в системе, сервер делает системный вызов `chroot()`, ограничивая все последующие операции начальным каталогом пользователя.

Все эти серверы доступны в виде пакетов или переносимых приложений, за исключением сервера BSD FTP, который входит в базовый комплект системы. Наличие большого количества серверов объясняется тем, что в программе WU-FTPD, которая долгое время широко использовалась, было выявлено много ошибок, поэтому независимые разработчики старались выпускать более безопасные варианты FTP-серверов. Стандартный FTP-сервер FreeBSD в этом отношении зарекомендовал себя достаточно хорошо. Программа ProFTPd имеет более широкие функциональные возможности. Другие серверы (включая те, что не указаны здесь; проверьте каталог `ftp` коллекции переносимых приложений или одноименный раздел меню пакетов программы `sysinstall`) могут быть интересны нестандартными особенностями, например расширенными средствами контроля доступа или возможностью обрабатывать передаваемые данные

Звпуск FTP-сервера

Как описывалось в главе 16, серверы могут вызываться из сценариев запуска системы или через суперсервер. В случае FTP-серверов, как правило, применяется второй вариант. Во FreeBSD файл `/etc/inetd.conf` содержит запись для запуска сервера BSD FTP, но по умолчанию она превращена в комментарий:

```
#ftp  stream  tcp  nowait  root  /usr/libexec/ftpd  ftpd  l
```

Чтобы активизировать FTP-сервер, выполните следующие действия

- удалите признак комментария (#);
- перезапустите демон `inetd` командой **`killall -SIGHUP inetd`**.

Теперь при обращении к FTP-серверу с другого компьютера будет выдаваться приглашение на регистрацию (предполагается, что FTP-порт не блокируется ни брандмауэром, ни каким либо другим средством)

```
$ ftp halrloprillalar
Connected to halrloprillalar.rodsbooks.com.
220 halrloprillalar.rodsbooks.com FTP server (Vers on 6 00LS) ready
Name (prill.rodsmith): rodsmith
331 Password required for rodsmith.
Password:
230 User rodsmith logged in
Remote system type is UNIX.
Using binary mode to transfer f les
ftp>
```

На заметку FTP клиенты рассматриваются в главе 24

В стандартной конфигурации требуется указывать имя пользователя и пароль Пользователи получают доступ к файлам в своих начальных каталогах, а также в других каталогах открытых для чтения (для всех файлов и каталогов сохраняется информация о правах доступа и владельцах) Это вполне разумная конфигурация но от нее приходится отказываться в случае анонимного FTP-сервера. Чтобы запустить такой сервер, создайте учетную запись `ftp` (см. главу 10). У нее должно быть минимум привилегий, и непосредственная регистрация в системе через эту учетную запись должна быть невозможна (поставьте символ '*' в поле пароля). Но ей необходим начальный каталог. FTP-сервер сделает его корневым для всех анонимных пользователей, чтобы они не могли получить доступ к системным файлам.

Для получения доступа к серверу в анонимном режиме пользователь вводит регистрационное имя `anonymous` и произвольный пароль (традиционно указывается почтовый адрес пользователя, но это не обязательно).

Альтернативные FTP-серверы запускаются сходным образом, только опции вызова будут другими. У некоторых серверов есть собственные конфигурационные файлы где задаются параметры запуска (параметры сервера BSD FTP указываются в конфигурационном файле демона `inetd`). Есть серверы, работающие автономно, без помощи демона `inetd` Ими удобно пользоваться в интенсивно эксплуатируемой системе.

NFS-сервер

Если необходимо, чтобы пользователи UNIX-систем могли осуществлять чтение и запись файлов на сервере непосредственно из приложений, сконфигурируйте сервер NFS Это подразумевает создание *экспортируемых файловых систем*, т е. каталогов

допускающих совместное использование. В определении такого каталога указывается, какие компьютеры могут обращаться к нему и каким способом (к примеру, определенным клиентам может быть разрешен доступ только для чтения). Далее нужно запустить сервер NFS, после чего клиенты смогут работать с экспортируемыми каталогами (конфигурирование клиентской части NFS рассматривается в главе 24).

Определение экспортируемых файловых систем

Для каждого экспортируемого каталога должно быть задано следующее: имя каталога, параметры, определяющие порядок обработки запросов сервером, а также имена или IP-адреса компьютеров, которым разрешен доступ к каталогу. Вся эта информация содержится в конфигурационном файле сервера NFS: `/etc/exports`. Каждому экспортируемому каталогу в этом файле соответствует одна строка (строк может быть несколько, если каталог экспортируется с разными параметрами разным клиентам). Строки, начинающиеся с символа '#', являются комментариями. Формат записей таков:

каталог [*параметры*] *список_клиентов*

В первом поле указывается полный путь к каталогу, например `/home` или `/usr/X11R6`. Спецификация каталога не должна содержать выражений '.' (текущий каталог) и '..' (родительский каталог). Кроме того, ни один из компонентов спецификации не должен являться символической ссылкой.

Параметры могут отсутствовать, но в некоторых случаях без них не обойтись. Наиболее распространенные параметры таковы:

- **`-alldirs`** — разрешает клиентам монтировать любой подкаталог указанного каталога. Например, если экспортируется каталог `/home` и существует подкаталог `/home/jennie`, то клиент сможет смонтировать `/home/jennie` вместо `/home`.
- **`maproot=`***пользователь*`[:`*группа1*`[:`*группа2*`[:...]]]`. По умолчанию пользователь `root` клиентской системы имеет очень ограниченные права доступа к файлам на сервере. Это служит мерой предосторожности на случай взлома клиентской системы. Опция `-maproot` позволяет задать пользователя и одну или несколько необязательных групп, которым соответствует локальный пользователь `root`. Например, если указать `-maproot=jennie`, пользователь `root` сможет обращаться к файлам на сервере от имени пользователя `jennie`.
- **`-mapall=`***пользователь*`[:`*группа1*`[:`*группа2*`[:...]]]` — эквивалент опции `-maproot`, но установки применяются ко *всем* пользователям клиентской системы. По умолчанию пользователям назначаются те же идентификаторы, что и в локальной системе, поэтому пользователь, например, с идентификатором 1002 получит доступ к тем же файлам, что и пользователь с аналогичным идентификатором на сервере.
- **`ro` или `-o`**. По умолчанию сервер NFS предоставляет клиентам полный доступ к экспортируемым каталогам. Ограничения реализуются с помощью обычных прав доступа к файлам и каталогам. Указанные опции заставляют сервер предоставлять доступ только для чтения.

Последний компонент записи представляет собой список клиентов, которым разрешен доступ к экспортируемому каталогу. Спецификации клиентов разделяются пробелами. Поддерживается несколько клиентских спецификаций:

- **Имена узлов**. Разрешается указывать имена компьютеров, например `nova.luna.edu` (или просто `nova`, если DNS-подсистема сервера ищет узлы в домене `luna.edu`).

- **IP-адреса**. Вместо имени компьютера может стоять IP адрес, например 172.17.2.251. В этом случае система будет работать чуть быстрее, поскольку не выполняется поиск в DNS.
- **Сетевые адреса**. Чтобы охватить все компьютеры сети, воспользуйтесь опцией `-network`, указав укороченный адрес сети. Можно также задать маску сети с помощью опции `netmask`. Например, спецификация `-network 172.17.2 netmask 255.255.255.0` заставляет сервер принимать запросы от всех клиентов в адресном диапазоне 172.17.2.0/24.
- **Сетевые группы NIS**. Если в сети функционирует сервер NIS, можно указать имя сетевой группы NIS. Вообще говоря, система всегда сначала пытается интерпретировать имя узла как имя сетевой группы.
- **Отсутствие спецификации**. Если клиентская спецификация отсутствует, сервер NFS принимает запросы от любых клиентов.

Отсутствие спецификации очень опасно! К такому результату может также привести опечатка или ошибка в клиентской спецификации, вследствие чего сервер окажется открытым для любых запросов. В качестве дополнительной меры предосторожности создайте правило брандмауэра (см. главу 17), ограничивающее доступ к порту сервера NFS (2049).

В качестве примера рассмотрим конфигурацию сервера, показанную в листинге 18.1. Здесь экспортируются два каталога `/usr/src` и `/home`. Доступ к первому из них разрешен компьютерам сети 172.17.2.0/24, а также компьютерам `nova`, `nebula`, `blackhole` и `browndwarf.luna.edu`. Обратная косая черта является символом продолжения строки. Дополнительные параметры отсутствуют. Каталог `/home` экспортируется с опцией `-alldirs` на компьютеры 172.17.4.8 `nova` и `nebula`. В случае компьютера `blackhole` параметры экспорта другие: пользователь `root` этого компьютера получит доступ к файлам сервера от имени пользователя `jennie`.

Листинг 18.1. Образец файла `/etc/exports`

```
/usr/src -network 172.17.2  netmask 255.255.255.0 nova nebula \
        blackhole browndwarf.luna.edu
home  -alldirs  172.17.4.8 nova nebula
home   maproot=jennie  blackhole
```

Вопросы безопасности

Необходимо следить за безопасностью сервера NFS. Модель надежных узлов означает, что сервер делегирует клиентам часть обязанностей по обеспечению безопасности. Если клиентский компьютер взломан, сервер оказывается под угрозой. Это не значит, что злоумышленник обязательно получит доступ на уровне суперпользователя, но ему станут доступны файлы других пользователей.

Прежде всего тщательно проверьте список надежных узлов в файле `/etc/exports`. Убедитесь в том, что для каждого экспортируемого каталога задан хотя бы один узел. В листинге 18.1 для наглядности указываются в основном имена компьютеров, но на практике этого следует избегать. Если злоумышленник взломал DNS-сервер, с которым взаимодействует сервер NFS, то доменные имена становятся ненадежным средством аутентификации. IP адреса сложнее подделать, хотя и в этом нет ничего невозможного, особенно если злоумышленник имеет физический доступ в сеть.

Желательно, чтобы система безопасности была многоуровневой. Ограничения в файле `/etc/exports` должны быть первым, но не единственным средством защиты.

Создайте дополнительные правила брандмауэра, чтобы ошибка в конфигурации файла `/etc/exports` не сделала систему открытой для атак. Уделите особое внимание правам доступа к файлам экспортируемых каталогов. По возможности используйте опцию `-maparoot`, чтобы весь доступ велся от имени локального пользователя с минимальными привилегиями.

Запуск NFS-сервера

Сервер NFS реализован в виде демона `nfsd`. Он обычно вызывается из сценариев запуска системы. В файле `/etc/rc.conf` должна быть такая строка:

```
nfs_server_enable="YES"
```

Если указанный параметр отсутствует или равен `NO`, задайте его должным образом и NFS-сервер будет запущен при следующей перезагрузке системы. Чтобы изменения вступили в силу немедленно, введите **`/etc/netstart`**. Если же система получает IP-адрес по протоколу DHCP, воспользуйтесь командой **`killall dhclient; /etc/netstart`**. Впрочем, серверы обычно работают со статическими IP-адресами, чтобы клиентам было проще находить их.

Samba

В настоящее время большинство компьютеров семейства x86 работает под управлением Microsoft Windows. FreeBSD-системы чаще всего используются в качестве серверов для клиентов Windows. Что касается обеспечения совместного доступа к файлам, то во FreeBSD необходимо включить поддержку протоколов SMB/CIFS. Это реализуется с помощью пакета Samba (`www.samba.org`). Samba — довольно сложный сервер, поэтому мы рассмотрим лишь основные его особенности.

Конфигурационный файл сервера Samba называется `smb.conf` и по умолчанию помещается в каталог `/usr/local/etc`. В этом файле есть раздел `global`, хранящий общие установки, и один или несколько разделов для *совместных ресурсов*. Таковыми могут быть монтируемые каталоги или сетевые принтеры. Названия разделов указываются в квадратных скобках. По общепринятому соглашению, строки, относящиеся к разделу, выделяются отступами. Символ '`#`' обозначает комментарий.

Настройка глобальных параметров

Стандартный файл `smb.conf` содержит большой раздел `global` с достаточно полными комментариями. Из них можно многое узнать о конфигурации сервера Samba. Более короткий вариант раздела может выглядеть так:

```
[global]
    workgroup= GALAXY
    netbios name = WORMHOLE
    server string = Local file server
    hosts allow = 172.17.2.
    load printers = Yes
    security = User
    encrypt passwords = Yes
```

Формат строк прост:

```
параметр = значение
```

Регистр символов не важен, если только в качестве значения не указано имя каталога. Как видно из показанного примера, пробелы разрешены как в названиях параметров, так и в их значениях. Назначение параметров таково:

- **workgroup** В протоколах SMB/CIFS (точнее в интерфейсе NetBIOS, на осно ве которого они реализованы) компьютеры объединяются в *рабочие группы До мен* представляет собой рабочую группу с центральным сервером аутентифика ции, называющимся *контроллером домена*. В параметре workgroup нужно указы вать название рабочей группы или домена. Если он отсутствует, сервер Samba не включается в список "Сетевое окружение" Windows-систем. Это одна из немно гих установок которую приходится менять в стандартной конфигурации
- **netbios name** — задает имя, на которое реагирует сервер Samba Оно может отличаться от сетевого имени TCP/IP, но во избежание путаницы лучше, что бы они совпадали. Если параметр отсутствует, используется имя компьютера TCP/IP без доменного компонента
- **server string** — задает описательную строку отображаемую в проводнике Windows

Внимание

В некоторых конфигурациях сервера описательная строка идентифи цирует не только компьютер, но и программное обеспечение сервера Это небезопасно, поскольку злоумышленник может получить инфор мацию о версии сервера, а значит, и его уязвимых местах В стандарт ном варианте файла параметр server string содержит строку Samba server, которой вполне достаточно

- **hosts allow** В протоколах SMB/CIFS аутентификация основана на имени пользователя и пароле, но сервер Samba дополнительно может ограничивать доступ на базе имен компьютеров, IP-адресов и сетевых адресов. В показанном примере доступ разрешен компьютерам сети 172.17 2.0/24 (хвостовые нули и сетевая маска подставляются сервером автоматически).
- **load printers**. Samba может функционировать в качестве сервера печати а не только файлового сервера Если параметр load printers равен Yes, сервер читает имена принтеров из файла /etc/printcap и создает каталоги совмест ного доступа для всех доступных принтеров. Необходимо также наличие специ ального раздела printers, о чем будет рассказываться в разделе "Определение каталогов совместного доступа
- **security**. Samba поддерживает несколько методов интерпретации имен поль зователей и паролей. Наиболее удобный из них — User, в котором сервер ищет имена в пользовательской базе данных FreeBSD, а пароли проверяются по стандартному файлу паролей или по базе паролей Samba в зависимости от значения параметра encrypt password
- **encrypt passwords**. Этот параметр заслуживает особого внимания Значение по умолчанию, равное No, заставляет сервер принимать от клиентов незашиф рованные пароли. Эти пароли сверяются со стандартным файлом паролей FreeBSD Проблема в том, что современные версии Windows (начиная с Windows 95 OSR2) передают только зашифрованные пароли. Чтобы сервер Samba мог работать с современными клиентами Windows нужно либо пере конфигурировать последние, либо разрешить шифрование паролей, как пока зано в разделе global. При этом будет создана отдельная база паролей Samba.

Обычно приходится менять параметры workgroup и encrypt passwords Осталь ные параметры содержат разумные значения по умолчанию или по крайней мере та кие значения, которые не делают сервер невидимым для клиентов Ряд параметров нужно менять, если компьютер является контроллером домена или сервером имен NetBIOS. Они описаны в документации к Samba

Настройка паролей

Для администраторов-новичков наиболее сложный аспект конфигурирования сервера Samba — настройка паролей. Как указывалось выше, все современные версии Windows сконфигурированы на передачу зашифрованных паролей. Это правильно с точки зрения безопасности, поскольку злоумышленник не сможет перехватить пароль с помощью сетевого анализатора пакетов. К сожалению, метод шифрования, применяемый в протоколах SMB/CIFS, несовместим с тем методом, который используется во FreeBSD для шифрования локальной базы паролей. Таким образом, приходится вести две базы данных: для обычных сеансов регистрации пользователей и для сеансов Samba.

Для манипулирования паролями Samba предназначена утилита `smbpasswd`. Вот как создается новая запись в базе данных:

```
# smbpasswd -a jennie
New SMB password:
Retype new SMB password:
unable to open passdb database
Added user jennie.
```

Опция `-a` заставляет утилиту добавить учетную запись пользователя в базу паролей. Этот пользователь должен быть уже описан в стандартной базе пользователей FreeBSD. Сообщение `unable to open passdb database` может появляться при первом вызове утилиты с опцией `-a`. Оно означает, что файл паролей (`/usr/local/private/smbpasswd`) не существует. Утилита `smbpasswd` создает его автоматически.

В пакет Samba входит стандартная база паролей `smbpasswd`, которая может вызывать проблемы. Если утилита `smbpasswd` не работает, попробуйте удалить этот файл. В противном случае удалите из него записи для различных системных учетных записей, таких как `root` или `daemon`.

После того как учетная запись создана, утилиту `smbpasswd` можно использовать для смены паролей, подобно стандартной команде `passwd`. Опцию `-a` в этом случае указывать не нужно. Утилита `smbpasswd` доступна рядовым пользователям.

Зашифрованные пароли удобны с точки зрения безопасности, но если есть уверенность в том, что локальная сеть безопасна, можно перейти на обычные пароли. Это позволит вести одну базу паролей, а не две. В таком случае нужно переконфигурировать клиентские Windows-системы. Соответствующие конфигурационные файлы находятся в каталоге `/usr/local/share/doc/samba/Registry` и носят имена вида *версия*`PlainPassword.reg`, где *версия* — это обозначение версии Windows, например `Win98` или `Wind2000`. Скопируйте соответствующий файл на FAT-дискету, вставьте ее в клиентский компьютер и выполните двойной щелчок на файле. В результате Windows обновит системный реестр. Повторите эту процедуру на всех клиентах Windows. Затем задайте параметр `encrypt passwords` равным `No` в файле `smb.conf` и забудьте о базе данных `smbpasswd`.

Определение каталогов совместного доступа

Помимо раздела `global` файл `smb.conf` содержит определения различных совместных ресурсов. Типичное определение выглядит так:

```
[hubble]
    path = /home/public/hubble images
    comment = Telescopic Images
```

```
b owseable = Yes
writeable = No
force user = currie
write list = grunsveld, linnehan, newman, massimino
```

Назначение использованных параметров таково.

- **path** Это наиболее важный параметр. Он задает каталог совместного доступа. В данном случае пользователи, монтирующие ресурс hubble, увидят содержимое каталога /home/public/hubble images. Значение по умолчанию — /tmp, что редко соответствует действительности.
- **comment** Как и в разделе global, этот параметр задает строку с описанием назначения ресурса, отображаемую проводником Windows. Этот параметр не является строго необходимым, но он желателен.
- **browseable** Windows-клиенты разрешают пользователям просматривать содержимое монтируемых ресурсов сервера. Этот параметр сообщает серверу о том, допустим ли просмотр ресурса. Значение No означает, что ресурс доступен, но пользователь будет вынужден самостоятельно указывать необходимые имена файлов.
- **writeable** Этот параметр определяет, доступен ли ресурс для записи. Значение по умолчанию — No, так что в данном примере строка избыточна. Есть также эквивалентный параметр write ok. Противоположный параметр называется read only.
- **force user** По умолчанию сервер Samba проверяет имя пользователя, предоставленное клиентом, чтобы определить, какие файлы доступны пользователю. Опция force user заставляет сервер осуществлять доступ к файлам от имени указанного пользователя.
- **write list** Этот параметр задает пользователей, которым разрешена запись в ресурс, даже если он предназначен только для чтения. Параметр write list не зависит от параметра force user: указанным пользователям в любом случае предоставляется право записи, но создаваемые ими файлы будут принадлежать пользователю currie (в данном примере это уполномоченный пользователь). Противоположный эффект имеет параметр read list: он позволяет запретить определенным пользователям запись в открытые каталоги.

Показанная конфигурация раздела подходит для каталогов совместного хранения документов или программ, которые не нужно инсталлировать на каждой рабочей станции. Но сервер Samba часто применяется для организации доступа к файлам отдельных пользователей. В этом случае идеально подходит раздел homes. Он отличается от остальных разделов тем, что параметр path динамически задается равным начальному каталогу текущего пользователя. Таким образом, определение параметра path включать не нужно. Например:

```
[homes]
    comment = Home Directories
    writeable = Yes
    browseable = No
    create mask = 0644
```

Как видите, это очень простое определение. В нем используется лишь один дополнительный параметр: create mask. Он задает максимально допустимые права доступа к создаваемым файлам. Значение по умолчанию — 0744. В проводнике Windows ресурсу homes соответствует имя пользователя. Параметр browseable, равный Yes, отменяет такую установку, делая ресурс доступным под именем homes. Обычно в этом нет необходимости, поэтому определение ресурса homes как правило включает строку browseable No.

Наличие раздела homes приводит к тому что каждый пользователь видит каталог, имя которого соответствует регистрационному имени самого пользователя Например, на рис. 18 1 присутствует папка rocksmith Отображаются ткже две ссылки на принтеры, создаваемые благодаря параметру load printers = Yes и специальному разделу printers, который является частью стандартного файла smb.conf. Определения принтеров напоминают определения каталогов совместного доступа, но дополнитель но содержат строку printable = Yes Для каждого принтера, определенного в файле /etc/printcap, допускается создавать ровно один раздел printers

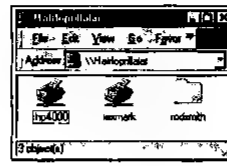


Рис 18.1 Совместные ресурсы Samba отображаются в проводнике Windows в виде обычных папок или принтеров

Запуск серверв Samba

Сервер Samba состоит из двух взаимосвязаиных демонов: smbd и nmbd. Первый из них это собственно файловый сервер Он обрабатывает запросы от клиентов, за прашивающих передачу файлов Демон nmbd просматривает содержимое ресурсов и выполняет преобразование имен Он выдает клиентам информацию которая отобра жается проводником Windows (см. рис. 18 1). Оба демона нужны для нормального функционирования сервера Samba.

Во FreeBSD в пакет Samba входит сценарий запуска называющийся /usr/local/etc/rc.d/samba.sh.sample. Он предназначен для вызова обоих демонов. Сценарий не выполняется автоматически на этапе начальной загрузки если только не убрать расширение .sample. Для однократного запуска сервера нужно просто ввести имя сценария в режиме суперпользователя Наличие выполняющегося сервера можно про верить с помощью команды ps

```
# ps ax | grep mbd
14925   ??   Is        0:00.01 /usr/local/sbin/smbd  D
14927   ??   Ss        0:00.06 /usr/local/sbin/nmbd  D
14955   p3   R+        0:00.00 grep mbd
```

Результаты работы команды передаются по каналу утилите grep, которая отбирает строки, содержащие подстроку "mbd" Если сервер Samba по какой-то причине отсут ствует или работает не так, как следует, просмотрите файл /var/log/messages В нем содержатся сообщения об ошибках, которые помогут установить причину. Сервер Samba создает также журнальные файлы /var/log/log.nmbd и /var/log/log.smbd для сообщений соответствующих демонов В них тоже содержится полезная отладочная информация.

Резюме

Файловые серверы — важные компоненты многих сетей, и FreeBSD — удобная платформа для запуска таких серверов. FTP является одним из старейших протоколов передачи файлов и по-прежнему широко применяется, особенно для обеспечения межплатформенного и анонимного доступа к файлам в Internet. Более сложные протоколы совместного использования файлов — NFS и SMB/CIFS — предназначены для обслуживания клиентов определенных операционных систем. Серверы NFS используются в среде UNIX и довольно просто конфигурируются. Пакет Samba реализует поддержку SMB/CIFS во FreeBSD.

Полный справочник по

Глава 19

Почтовые серверы

Электронная почта — важное средство общения в современном мире. Она позволяет не только вести переписку с друзьями и родственниками, но и обмениваться деловыми документами. Вот почему глобальные каналы передачи почты должны быть надежными. А поскольку надежность – отличительная черта FreeBSD (и UNIX систем вообще), неудивительно, что почтовые сообщения на пути к адресату проходят хотя бы одну UNIX систему. FreeBSD является удобной платформой для серверов, обрабатывающих почту.

В этой главе мы сначала поговорим о том, как организуется распространение почты в Internet, сделав акцент на различных типах серверов. Затем будет приведена информация о конфигурировании основных почтовых серверов. В конце главы рассматривается программа Fetchmail, которая не является сервером, но играет роль "связующего звена" между серверами в определенных сценариях доставки почты.

Типы почтовых серверов

Во многих организациях один из компьютеров функционирует в качестве почтового сервера. В действительности на нем может быть запущено несколько серверов, отвечающих за обработку различных почтовых протоколов. Некоторые из этих протоколов (известны как *протоколы принудительной доставки*) требуют, чтобы передачу почты инициировал отправитель. В других протоколах (их называют *протоколами доставки по запросу*) начальный запрос поступает от получателя. Для того чтобы грамотно сконфигурировать серверный компьютер, нужно понимать, как взаимодействуют эти протоколы.

Серверы принудительной доставки почты и серверы доставки по запросу

Почтовые серверы могут пересылать почту между пользователями одного компьютера, но более интересная (и сложная) задача – доставка почты на другие компьютеры. Важную роль в этом играет *ретранслятор почты* — система, принимающая почту от одного компьютера и посылающая ее на другой. На пути к адресату сообщение может пройти через несколько ретрансляторов.

Большинство ретрансляторов почты работает по протоколам принудительной доставки, в которых пересылку почты инициирует отправитель. Например, когда пользователь составляет письмо, почтовый агент направляет это письмо на сервер исходящей почты (расположенный в локальной сети или у провайдера) по протоколу принудительной доставки, в качестве которого, как правило, выступает SMTP (Simple Mail Transport Protocol – простой протокол передачи электронной почты). Этот сервер использует протокол SMTP для доставки сообщения следующему почтовому серверу и т.д. Важной особенностью протоколов данного класса является то, что компьютер-получатель должен быть постоянно доступен. Если отправитель не может связаться с получателем, почта может быть утеряна (SMTP-серверы сохраняют почту в случае сбоя и пытаются доставить ее повторно через определенное время, но рано или поздно недоставленное сообщение будет удалено).

Последнее звено в цепи доставки почты обычно составляют серверы доставки по запросу. Наиболее популярные протоколы данного класса — POP (Post Office Protocol - почтовый протокол) и IMAP (Internet Message Access Protocol — протокол доступа к сообщениям в сети Internet). Они применяются, когда конечным получателем является рабочая станция, на которой не запущен сервер принудительной достав-

ки почты. Как правило, рабочие станции подключаются к Internet нерегулярно, по протоколу PPP. В этом случае запускать на них SMTP-серверы бессмысленно.

Типичная цепочка доставки почты изображена на рис. 19.1. В этом примере пользователь рабочей станции `marconi.pangaea.edu` отправляет письмо пользователю домена `threeroomco.com` — скажем, `ben@threeroomco.com`. Почтовый агент рабочей станции `marconi` должен быть сконфигурирован на работу с правильным сервером исходящей почты, в данном случае — `morse.pangaea.edu`. Этот сервер может доставлять почту непосредственно в нужный домен или пользоваться услугами другого сервера как ретранслятора. На рис. 19.1 компьютер `tesla.pangaea.edu` является последним ретранслятором домена `pangaea.edu`.

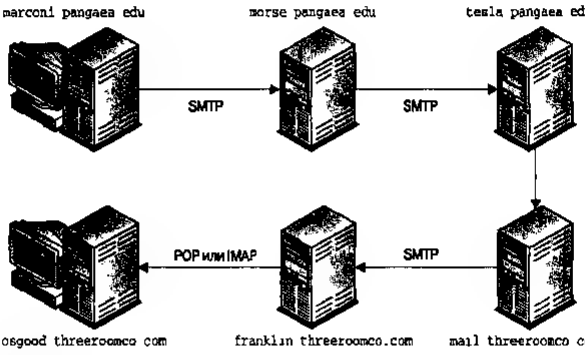


Рис. 19.1. Почтовое сообщение часто проходит через целый ряд компьютеров прежде чем попасть к адресату

Сервер исходящей почты должен определить адрес получателя. Для этого выполняется обычный поиск в DNS, но есть один нюанс: в DNS поддерживается специальный тип адресных записей, используемых исключительно в целях доставки почты. Они называются MX (от "mail exchange" – обмен почтой). Записи MX могут быть связаны с отдельными компьютерами, но обычно они закрепляются за целыми доменами. Каждая такая запись ссылается на конкретный компьютер, расположенный в этом или другом домене. В нашем примере почта направляется по адресу `ben@threeroomco.com`, поэтому компьютер `tesla.pangaea.edu` ищет запись MX для домена `threeroomco.com`. Эта запись ссылается на компьютер `mail.threeroomco.com`, куда и пересылается сообщение. Как и в случае исходящей почты, сервер, принимающий почту из внешней сети, не обязательно является последним звеном в цепи. Он может переадресовывать почту другому внутреннему серверу, как, например `franklin.threeroomco.com` на рис. 19.1. На этом компьютере работают *два* почтовых сервера: один — по протоколу SMTP (принудительная доставка), другой — по протоколу POP или IMAP (доставка по запросу). Конечный получатель (`ben`) сидит за рабочей станцией (`osgood.threeroomco.com`) и использует программу чтения почты для доступа к своему почтовому ящику, хранящемуся на компьютере `franklin.threeroomco.com`.

На заметку

Использование протокола доставки по запросу не является обязательным. Получатель может запустить программу чтения почты непосредственно на компьютере `franklin.threeroomco.com`, используя протокол дистанционной регистрации (глава 21).

В рассмотренном примере протокол доставки по запросу используется лишь на самом последнем этапе. Это типичная конфигурация, поскольку такие протоколы удобны, когда компьютер-получатель подключен к Internet непостоянно. В разделе "Конфигурирование программы Fetchmail" будет показано, как использовать протокол доставки по запросу в середине цепочки, а не в конце. Такое применение протокола может быть необходимо, если FreeBSD-система является почтовым сервером только для локальной сети или для небольшой сети, скрытой за маршрутизатором NAT (см. главу 17).

Следует отметить, что маршрут доставки почты не всегда выглядит так, как изображено на рис. 19.1. В доменах `pangaea.edu` и `threeroomco.com` может быть больше или меньше ретрансляторов почты, да и между самими доменами могут находиться дополнительные ретрансляторы. Детали зависят от особенностей конкретных доменов. Дополнительные серверы используются для раздельного контроля входящей и исходящей почты, распределения нагрузки и преобразования почты между различными форматами.

Серверы принудительной доставки почты

Серверы принудительной доставки — чрезвычайно важные компоненты в системе обмена электронной почтой. В основном все они работают по протоколу SMTP. Для FreeBSD доступно множество таких серверов. Они обладают чрезвычайно мощными функциональными возможностями и способны удовлетворить потребности даже очень крупных доменов. Перечислим наиболее популярные серверы:

- **sendmail** (`www.sendmail.org`). Это наиболее распространенный из существующих SMTP-серверов. Согласно исследованиям, программа `sendmail` установлена почти на половине почтовых серверов Internet. Это также стандартный почтовый сервер FreeBSD, который инсталлируется по умолчанию.
- **qmail** (`www.qmail.org`). Этот сервер реализован модульным способом в целях повышения эффективности и безопасности и является популярной альтернативой программе `sendmail`. Он, впрочем, использует другой формат почтовых файлов, поэтому существующие почтовые программы, включая серверы доставки по запросу, придется заменить или переконфигурировать.
- **Exim** (`www.exim.org`). Эта программа использует тот же формат почтовых файлов, что и `sendmail`, и содержит необычайно мощные средства фильтрации почты, основанные на поиске образцов в заголовках и теле сообщений. Расширенные механизмы фильтрации очень удобны для борьбы со спамом.
- **Postfix** (`www.postfix.org`). Как и `qmail`, программа Postfix реализована модульным способом, что способствует повышению безопасности и производительности. Формат почтовых файлов такой же, как и у `sendmail`.

Все эти программы есть в коллекции переносимых приложений FreeBSD, а `sendmail` и Exim доступны в виде пакетов (способы инсталляции таких программ описывались в главе 11). Поскольку `sendmail` инсталлируется по умолчанию и является наиболее популярным почтовым сервером Internet, именно эта программа будет описана в разделе "Конфигурирование сервера SMTP".

Серверы доставки почты по запросу

Серверы доставки по запросу гораздо меньше по размеру и проще в реализации чем серверы принудительной доставки почты Их функциональные возможности бо лее ограничены, поэтому такие серверы заслуживают гораздо меньше внимания Тем не менее существует ряд аспектов, которые необходимо учесть

В первую очередь необходимо выбрать используемый протокол. Наиболее попу лярны протоколы POP и IMAP Первый из них проще: он дает возможность пользо вателям загружать почту с сервера, но не позволяет организовывать хранение почты на сервере Идея заключается в том, что пользователь создает папки для хранения почты на своем компьютере с помощью почтового клиента В отличие от этого более сложный протокол IMAP разрешает создавать папки на сервере и доставлять лишь заголовки сообщений Такой подход предъявляет очень высокие требования к жест кому диску сервера, зато упрощает задачу пользователям, которым часто приходится запускать различные почтовые клиенты (например на рабочей станции и ноутбуке), ведь в конечном итоге почта не привязывается к конкретному клиенту Протоколы POP и IMAP не являются взаимоисключающими и могут применяться одновременно

Серверы доставки почты по запросу могут работать с почтовыми клиентами любой операционной системы Windows, Mac OS, FreeBSD и т д. Клиент, поддерживающий тот или иной протокол, будет взаимодействовать с любым сервером этого протокола. Ниже перечислены серверы такого типа

- **Cyrus IMAP** (`http://asq.web cmu.edu/cyrus/`). Этот сервер поддерживает протоколы POP и IMAP. Для папок IMAP используется собственный формат хранения
- **UW IMAP** (`http://www.washington.edu/imap/`). В пакет Вашингтонского университета входят серверы POP и IMAP Папки IMAP хранятся в начальных каталогах пользователей.
- **dkimap** (`http://freshmeat net/projects/dkimap/`) Это небольшой сервер IMAP
- **QPopper** (`http://www.eudora.com/qpopper/`). Этот POP сервер раньше был коммерческим продуктом, но теперь распространяется с открытым исходным кодом
- **qmail-pop3d** Эта программа поставляется вместе с SMTP сервером `qmail` и умеет обрабатывать файлы соответствующего формата.

Для небольшой организации подойдет любой из этих серверов Все они доступны либо в виде пакетов, либо в виде переносимых приложений.

Имеет также значение *версия* протокола В настоящее время наиболее популярны протоколы POP-3 и IMAP-4 Если используются устаревшие почтовые клиенты, при дется поискать для них серверы, поддерживающие более старые протоколы, например POP-2 (он работает с другим портом) Существуют также защищенные варианты от дельных протоколов, реализующие шифрование паролей и данных

Конфигурирование сервера SMTP

Конфигурация SMTP-сераера может быть довольно простой или чрезвычайно сложной Все зависит от потребностей конкретной системы В идеальном случае стан дартная конфигурация программы `sendmail` потребует минимальных изменений Но иногда приходится приложить немало усилий, прежде чем программа заработает должным образом В этом разделе мы рассмотрим лишь основные аспекты конфигу рирования `sendmail`

Конфигурирование домена

В первую очередь необходимо сконфигурировать текущий домен Как указывалось выше, DNS позволяет закрепить за доменом почтовый сервер. Этим должен заниматься администратор DNS-сервера В файле конфигурации домена должна присутствовать запись следующего вида

```
@        IN   MX     10   mail.three.comco.com.
```

Конфигурационный файл обычно находится в каталоге `/var/named` и называется в соответствии с именем домена, например `named.theeroomco.com` Дополнительная информация о конфигурировании DNS-сервера, включая объяснение полей конфигурационного файла, приведена в главе 22

На заметку Можно сконфигурировать домен так, чтобы его почту обрабатывал почтовый сервер другого домена. Эта конфигурация удобна, если почта хранится на сервере провайдера

Конфигурационные файлы sendmail

Параметры конфигурации программы `sendmail` хранятся в файле `/etc/mail/sendmail.cf`. К сожалению, его формат слишком сложен для администраторов-новичков Даже опытным администраторам не рекомендуется редактировать этот файл, поскольку единственная опечатка способна вывести сервер из строя Вместо этого редактируйте файл макросов `/etc/mail/freebsd.mc` Его формат гораздо понятнее, чем у файла `sendmail.cf` По окончании редактирования нужно выполнить преобразование форматов:

```
# cd /usr/share/sendmail/cf/m4
# m4 cf.m4 /etc/mail/freebsd.mc > /etc/mail/sendmail.cf
```

Внимание Перед редактированием файла `freebsd.mc` создайте его резервную копию или же скопируйте файл под другим именем, отредактируйте копию и подставьте ее имя в показанную выше команду Точно так же нужно создать резервную копию файла `sendmail.cf` Это позволит вернуться к стандартной конфигурации в случае необходимости.

После того как новый файл `sendmail.cf` создан заставьте программу `sendmail` прочитать его

```
# killall -SIGHUP sendmail
```

Изменение стандартных параметров

Стандартная конфигурация программы `sendmail` вполне работоспособна По умолчанию программа принимает почту адресованную только пользователям текущего компьютера Если компьютер является почтовым сервером домена, эту конфигурацию придется изменить. Что касается отправки почты, то по умолчанию программа доставляет почту непосредственно заданному компьютеру Для почтового сервера такая конфигурация приемлема, но иногда почту нужно ретранслировать через другой сервер, как, например, `morse.pangaea.edu` на рис 19.1

Задание компьютеров для приема почты

Рассмотрим еще раз рис 19.1 Каким образом компьютеры в цепочке определяют, когда нужно послать сообщение на другой компьютер и когда сохранить это сообще

ние в локальной почтовой очереди? В стандартной конфигурации sendmail ответ на этот вопрос дает имя компьютера, указанное в адресе письма. Когда компьютер franklin.threeroomco.com, например, получает сообщение, адресованное пользователю ben@franklin.threeroomco.com, он "понимает", что оно предназначено локальной учетной записи ben. Но в примере на рис. 19.1 письмо *не* адресовано пользователю ben@franklin.threeroomco.com — оно направляется по адресу ben@threeroomco.com. По умолчанию программа sendmail отвергнет его, поскольку имя компьютера не равно threeroomco.com, хотя он и находится в этом домене.

Проблема решается с помощью файла /etc/mail/local-host-names (в стандартной инсталляции он не существует, поэтому его придется создать). В нем содержатся имена компьютеров, которые программа sendmail интерпретирует как локальные. Например, чтобы программа принимала почту, адресованную пользователям домена threeroomco.com, имя этого домена должно быть включено в файл local-host-names. Имя каждого поддерживаемого домена задается в отдельной строке, например:

```
tworoomco.com
threeroomco.com
fourroomco.com
luna.edu
```

Естественно, необходимо должным образом сконфигурировать DNS-серверы этих доменов, чтобы их записи MX ссылались на данный почтовый сервер.

Имя /etc/mail/local-host-names не задано жестко в исходном коде. Оно указано в одной из инструкций define в файле /etc/mail/freebsd.mc. Поищите строку, в которой определяется переменная confCW_FILE.

Если пользователь попытается послать письмо в домен, не являющийся локальным, программа sendmail воспримет это как попытку ретрансляции. В зависимости от настроек ретрансляции и прочих факторов, таких как IP-адрес отправителя, сервер может либо отвергнуть письмо, либо принять его и передать конечному получателю.

Задание шлюза исходящей почты

Когда программа получает почту, адресованную удаленной системе, она ищет ее доменное имя или запись MX и пытается доставить почту непосредственно компьютеру-адресату. Такой подход не всегда практичен. Многие провайдеры блокируют прямые исходящие SMTP-соединения в целях борьбы со спамом. Даже если этого ограничения нет, но компьютер имеет коммутируемый доступ в Internet, то прямая доставка почты неудобна тем, что в случае недоступности получателя программа поставит сообщение в очередь. И если при следующей обработке очереди компьютер окажется отключен от сети, сообщение может быть удалено. Иногда прямая доставка почты нежелательна, поскольку необходимо, чтобы вся почта проходила через один центральный сервер, например в целях учета или безопасности.

В подобных случаях нужно сконфигурировать программу sendmail на клиентских компьютерах таким образом, чтобы почта ретранслировалась через заданный сервер. Сделать это несложно. Введите в файле freebsd.mc строку следующего вида:

```
define(`SMART_HOST', `tesla.pangaea.edu')
```

Обратите внимание на два типа кавычек. Открывающая кавычка является обратной (`), а закрывающая — прямой ( ). Именно так их и нужно вводить, иначе строка будет интерпретирована неправильно.

Эта строка заставляет программу sendmail направлять всю исходящую почту компьютеру tesla.pangaea.edu, который отвечает за ее последующую доставку Система, сконфигурированная подобным образом, будет самостоятельно обрабатывать локальную почту В стандартном файле freebsd.mc такая строка является комментарием т е она начинается с ключевого слова dnl Удалите его и задайте нужное имя шлюза после чего введите **killall -SIGHUP sendmail**

Борьба со спамом

Несанкционированная массовая рассылка почты, или *спам*, — бич администраторов электронной почты На обработку спама тратятся ценные системные ресурсы, к тому же он раздражает пользователей В виде спама рассылаются различные рекламные сообщения, часто оскорбительного характера, которые необходимо блокировать Все современные почтовые серверы содержат средства борьбы со спамом, которых так много, что в рамках одной главы не представляется возможным их рассмотреть. Мы сделаем лишь краткий обзор доступных средств. Дополнительную информацию можно получить на специализированных Web-узлах www.mail abuse org spam.abuse.net и www.cauce.org

Спам бывает двух видов: входящий и исходящий Пользователей обычно беспокоит лишь входящий спам, но блокирование исходящего спама не менее важно если вследствие неправильной конфигурации почтовый сервер разрешает открытую ретрансляцию спама, администратора начнут заваливать жалобами Почтовые серверы могут вообще отказаться принимать почту от него

В последние годы серьезной проблемой стали почтовые вирусы "черви". Некоторые методики борьбы со спамом например фильтрация заголовков и содержимого сообщений эффективны и против "червей"

Блокирование входящего спама

Существует несколько методик выявления входящего спама В их основе лежит использование черных списков, фильтрация заголовков и фильтрация содержимого Спам можно просто удалять или же помещать в специальную папку, где пользователь сможет просмотреть его (по истечении определенного времени сообщения из этой папки автоматически удаляются)

Ни одна из методик борьбы со спамом не является совершенной. Сколько правил фильтрации ни задавай, все равно некоторые сообщения "прорываются" сквозь преграды. Не исключено, что наряду со спамом фильтрации подвергнется и легитимная почта Придется учиться на собственном опыте.

Черные списки

Черный список — это база данных, содержащая IP-адреса компьютеров, почту от которых нужно удалять Существуют различные критерии включения в такой список. Это могут быть компьютеры, рассылающие спам, либо неправильно сконфигурированные серверы, *допускающие* рассылку спама В табл 19.1 перечислены наиболее известные черные списки

Таблица 19.1. Распространенные черные списки

Название	Web-узел	Адрес сервера	Описание
RBL (Realtime Blackhole List)	`http://mail-abuse.org/rbl/`	`blackholes.mail-abuse.org`	В этом списке перечислены системы, которые рассылают спам, являются открытыми ретрансляторами или поддерживают спамеров
RSS (Relay Spam Stopper)	`http://www.mail-abuse.org/rss/`	`relays.mail-abuse.org`	В этом списке перечислены узлы открытой ретрансляции спама
DUL (Dial Up List)	`http://www.mail-abuse.org/dul/`	`dialups.mail-abuse.org`	В этом списке перечислены IP-адреса, связанные с коммутируемыми PPP-соединениями, поскольку такие пользователи должны работать с почтовыми серверами провайдеров
ORDB (Open Relay Database)	`http://www.ordb.org`	`relays.ordb.org`	Это аналог списка RSS с несколько иными критериями включения
RFC Ignorant	`http://www.rfc-ignorant.org`	Меняется; см. на Web-узле	В этих списках указаны системы, владельцы которых не соблюдают стандарты Internet, например не регистрируют корректную контактную информацию

Чтобы подключить черный список к программе `sendmail`, добавьте строку следующего вида в файл `freebsd.mc`:

```
FEATURE(dnsbl, `dialups.mail-abuse.org', `"550 Mail from dial up% rejected; see http://mail-abuse.org/dul/enduser.html"')
```

Подставьте нужный адрес сервера вместо `dialups.mail-abuse.org`. Можно также поменять текст сообщения об ошибке, возвращаемого отправителю блокируемого письма. Наличие такого сообщения важно, поскольку правила фильтрации могут препятствовать распространению легитимной почты, и пользователи должны знать об этом.

Черные списки, составляемые MAPS (Mail Abuse Prevention System — служба предотвращения почтовых злоупотреблений), а именно RBL, RSS и DUL, распространяются по подписке, и за их использование может взиматься плата. Есть списки, доступные для всех желающих.

Преимуществом черных списков является то, что сервер проверяет их в самом начале, еще до того, как загружено тело сообщения. Это важно для крупных почтовых серверов, поскольку снижаются требования к пропускной способности сервера. Недостаток же заключается в том, что "вина" сообщений определяется их IP-адресом. Если спамер взламывает систему и своей деятельностью обеспечивает ее попадание в черные списки, страдают легитимные пользователи системы.

Совет Одна из причин, по которой необходимо задавать шлюз исходящей почты, заключается в том, что адреса PPP-соединений включаются в списки типа DUL из соображений безопасности, а не потому, что тот или иной адрес используется спамерами. Таким образом, ретрансляция почты через сервер провайдера позволяет избежать ее блокирования.

Фильтрация заголовков

Другой подход к борьбе со спамом заключается в создании фильтров основанных на анализе заголовков сообщений. Заголовки — это начальные строки письма, содержащие служебную информацию, такую как адреса отправителя и получателя, тема сообщения и т.д. Фильтр может проверять, к примеру, наличие более пяти последовательных пробелов в поле темы Спамеры часто используют описательную тематическую строку в конце которой после группы пробелов указан числовой код. Это позволяет им обмануть фильтры, блокирующие поток сообщений с одинаковой тематической строкой

Программа `sendmail` содержит не так много средств проверки заголовков: разрешается фильтровать сообщения на основании имени или IP-адреса отправителя Более сложные фильтры реализуются с помощью вспомогательных утилит, как будет показано далее Если нужно включить проверку заголовков, создайте файл `/etc/mail/access` Во FreeBSD есть файл-образец `/etc/mail/access.sample`, который можно взять за основу Содержимое файла будет выглядеть примерно так:

```
badspammer.net              550 No spam allowed
antispam.badspammer.net     OK
spamsrus.com                REJECT
```

Коды `550` и `REJECT` заставляют программу отвергать почту полученную из указанных доменов. В первом случае задается также сообщение об ошибке, возвращаемое отправителю (`No spam allowed`). Код `OK` означает, что почта разрешена Это позволяет создавать исключения из правил В данном примере почта от узла `antispam.badspammer.net` принимается даже несмотря на то, что он относится к домену `badspammer.net`

Когда правила будут готовы преобразуйте файл `/etc/mail/access` в бинарный формат

```
# makemap hash /etc/mail/access.db < /etc/mail/access
```

Далее нужно заставить программу `sendmail` обновить свою конфигурацию как было показано на с 381.

Фильтры заголовков позволяют почтовому серверу отклонять сообщения на ранних этапах обработки. Серверы, имеющие расширенные средства фильтрации, например Exim и Postfix, могут проверять произвольные заголовки включая поле темы и заголовки, создаваемые программным обеспечением спамеров Такие фильтры подвержены тем же проблемам, что и черные списки В частности, можно легко заблокировать почту легитимных пользователей, если их провайдеры в недостаточной степени борются со спамом Кроме того, на проверку правил фильтрации уходит время

Фильтрация содержимого

Фильтры содержимого проверяют не только заголовки, но и тело письма Например, можно создать фильтр, который удаляет почту, содержащую упоминание S.1618 (отмененный закон США, на который часто ссылаются спамеры) Такие фильтры часто создаются с помощью утилиты `procmail`, которая вызывается программой `sendmail` для обработки почты, принятой сервером Утилита `procmail` помещает почтовые сообщения в почтовые папки отдельных пользователей и может принимать, отклонять и модифицировать сообщения на основании их содержимого. Некоторые почтовые серверы, например Exim, фильтруют содержимое без помощи внешних пакетов

В стандартной инсталляции `sendmail` утилита `procmail` не используется для доставки почты Эту утилиту нужно сначала инсталлировать, как описывалось в главе 11. Затем пользователь должен отредактировать файл `.forward` в начальном каталоге, включив в него следующую строку (обратите внимание на наличие кавычек)

```
"|exec /usr/local/bin/procmail || exit 75"
```

Далее необходимо задать в файле .procmailrc правила обработки почты. Они могут быть достаточно сложными, мы же рассмотрим два простых примера:

```
:0
*^Subject:.*       *
/dev/null

 0 B
*^.*mailto: *subject=3Dremove
/dev/null
```

Оба правила начинаются со строки :0. Во втором случае указан также код B, обозначающий поиск в теле сообщения, а не в заголовках. Каждое правило содержит одну или несколько строк, начинающихся со звездочки ('*'). Они задают условия срабатывания правила. В основном здесь указываются фрагменты текста письма, хотя некоторые символы имеют особое значение. Например, символ '^' обозначает начало строки. Выражению '.*' соответствует строка символов произвольной длины (в том числе пустая). Таким образом, первое правило описывает заголовок Subject (тематическая строка), содержащий, среди прочего, пять последовательных пробелов. Второму правилу соответствует строка mailto:, после которой встречается выражение subject=3Dremove (подобное характерно для ссылок типа "remove yourself from...", которые почти не встречаются в обычных письмах). Последняя строка каждого правила задает файл, в который помещается сообщение. В обоих случаях это /dev/null, т.е. сообщения удаляются. Если нужно направить сообщение другой программе, укажите ее имя после вертикальной черты ('|').

Блокирование исходящего спама

Исходящий спам должен блокироваться в не меньшей степени, чем входящий. Существуют два потенциальных источника спама.

- **Локальные пользователи.** Пользователи системы могут намеренно распространять спам. В любой организации должны существовать правила пресечения спама, и пользователи должны быть оповещены о них, чтобы в случае злоупотреблений можно было применять к нарушителям соответствующие меры.
- **Ретранслируемая почта.** Если вследствие неправильной конфигурации компьютер ретранслирует почту от несанкционированных отправителей, этим могут воспользоваться спамеры. Ретрансляция позволяет им, во-первых, скрывать свое местонахождение, а во вторых — перекладывать нагрузку по рассылке спама на сторонние серверы.

В первом случае все упирается в разработку и внедрение соглашений по пресечению распространения спама. Объясните пользователям, что почта, посылаемая более чем десяти получателям, уже вызывает подозрение. Спам — это коммерческое явление, поэтому пользователи должны с особой осторожностью подходить к рассылке рекламных объявлений и анонсов коммерческих продуктов и услуг. Но даже сообщения некоммерческого характера воспринимаются как спам, если они поступают от незнакомого отправителя без предварительного запроса со стороны получателя.

Возможность ретрансляции ограничивается различными конфигурационными параметрами программы sendmail. Стандартный файл freebsd.mc содержит такую установку:

```
FEATURE(relay_based_on_MX)
```

Эта строка заставляет программу ретранслировать почту от любого компьютера, находящегося в домене, DNS-запись MX которого ссылается на текущий почтовый сервер. Такая конфигурация удобна, если сервер является шлюзом многих доменов, но ею легко злоупотребить: спамеру достаточно будет модифицировать DNS-записи

домена, находящегося под его административным контролем Желательно вместо пре дыдущей установки задать следующую

```
FEATURE(relay_entire_domain)
```

В этом случае программа будет ретранслировать почту домена, в котором она работает. Такая конфигурация подходит для почтового сервера небольшой сети Еще один вариант — не включать ни одну из опций ретрансляции, а воспользоваться описанным выше файлом access Код RELAY позволяет явно указать компьютеры, которым разрешена ретрансляция:

```
172.29.39       RELAY
pangaea.edu     RELAY
```

В данном случае почта принимается от компьютеров домена pangaea edu и сети 172 29.39 0/24. Как правило, это оптимальный вариант контроля ретрансляции. Опции ретрансляции, задаваемые в конфигурационном файле программы sendmail, в луч шем случае рискованны

Одна из установок — FEATURE(promiscuous_relay) — заслуживает особого упоминания. Если она присутствует в конфигурационном файле, немедленно удалите ее, поскольку она включает режим откры той ретрансляции. Такие конфигурации являются наиболее желанным объектом "охоты" спамеров

Дополнительную информацию о настройке механизма ретрансляции и правилах борьбы со спамом можно найти по адресу http://www.mail-abuse.org/tsi/. Там приведены советы, касающиеся конфигурирования как программы sendmail, так и других серверов Проверьте, не поддерживает ли ваш почтовый сервер открытую ретрансляцию. Для этого нужно установить Telnet-соединение с сервером relay-test.mail-abuse.org, который в свою очередь, установит SMTP-соединение с запрашивающей стороной и выполнит серию тестов

Конфигурирование сервера POP или IMAP

Программа sendmail выполняет две функции. она принимает почту от внешних компьютеров и пересылает почту из локальной сети во внешние сети. Серверы доставки почты по запросу нужны для того, чтобы пользователи могли загружать почту с сервера, на котором работает программа sendmail Конфигурация таких серверов очень проста. Лишь очень крупные почтовые серверы нуждаются в более сложной настройке, чем та, что рассматривается в этом разделе

Ниже описывается конфигурация пакета UW IMAP (http.//www.washington edu/imap/), в который входят серверы POP и IMAP. Этот пакет доступен в коллек ции переносимых приложений (раздел mail/imap-iw)

После инсталляции пакета нужно переконфигурировать существующий файл /etc/inetd.conf, включив в него вызов сервера POP или IMAP. В стандартном варианте файла соответствующие строки превращены в комментарии, но в них демон стрируется вызов других исполняемых файлов. Вот как организуется вызов серверов POP-3 и IMAP-4

```
pop3  stream  tcp  nowait  root  /usr/local/libexec/ipop3d  ipop3d
imap4 stream  tcp  nowait  root  /usr/local/libexec/imapd   imapd
```

Серверы UW IMAP, как и другие серверы FreeBSD, осуществляющие аутентификацию на основании имен пользователей и паролей, взаимодействуют с базой паролей посредством модулей PAM. Чтобы этот механизм заработал, нужно включить приведенные ниже строки в файл /etc/pam conf (во FreeBSD 4 5 и более ранних версий)

```
imap    auth      required        pam_unix.so
imap    account   required        pam_unix.so     try_first_pass
imap    session   required        pam_deny.so
pop     auth      required        pam_unix.so
pop     account   required        pam_unix.so     try_first_pass
pop     session   required        pam_deny.so
```

Во FreeBSD 5.0 и выше следует создать файлы `imap` и `pop3` в каталоге `/etc/pam.d`. Они должны выглядеть так же, только первый столбец не нужен.

Далее необходимо заставить демон `inetd` обновить свою конфигурацию.

```
# killall -SIGHUP inetd
```

С этого момента серверы станут доступны почтовым клиентам. Воспользуйтесь любым POP- или IMAP-клиентом, чтобы подключиться к серверу и загрузить почту с него. В случае протокола IMAP почтовый клиент должен позволять перемещение почты между папками на сервере.

Сервер UW IMAP хранит почту в начальном каталоге пользователя. Следовательно, если учетная запись используется не только для обработки почты, клиенту IMAP понадобится немало времени на сканирование всех имеющихся файлов и каталогов. Некоторые из каталогов могут восприниматься как почтовые, хотя это не так. Другие серверы IMAP используют иные форматы хранения почтовых папок.

Конфигурирование программы Fetchmail

Как правило, протоколы доставки почты по запросу, в частности POP или IMAP, используются на самом последнем этапе. Они позволяют почтовым клиентам, таким как Outlook, Eudora или Sylpheed, загружать почту с сервера, на котором у пользователя есть учетная запись. Но бывают случаи, когда цепочка доставки почты расширяется: требуется загружать почту с внешнего сервера и помещать ее в почтовую очередь внутреннего сервера. Это дает возможность объединять почту нескольких серверов, а также манипулировать почтой, например запускать в локальной сети сервер IMAP даже если провайдер поддерживает только протокол POP.

Для расширения цепочки доставки нужен специальный почтовый клиент. Вместо того чтобы функционировать в качестве обычной программы чтения почты, он автоматически вставляет сообщения в почтовую очередь локальной или удаленной системы. Система-получатель дальше пересылает эту почту через SMTP-соединение или делает ее доступной для собственного сервера доставки по запросу. Популярной программой такого рода является Fetchmail (`http://www.catb.org/~esr/fetchmail/`), которая доступна как в виде пакета, так и в виде переносимого приложения.

Конфигурационный файл

В отличие от многих серверов, программе Fetchmail не нужны специальные привилегии. Она функционирует как обычный почтовый клиент, поэтому ее запускают от имени рядового пользователя или специального пользователя с минимальными привилегиями.

Конфигурационный файл программы называется `.fetchmailrc` и находится в начальном каталоге пользователя Fetchmail. В этом файле содержится вся информация, необходимая для работы программы.

- адрес сервера доставки по запросу, с которого загружается почта;
- имя пользователя, пароль и протокол, используемые для загрузки почты.

- адрес сервера принудительной доставки с которым осуществляется взаимодействие, или имя программы, запускаемой для доставки загруженной почты;
- почтовый адрес, по которому высылается доставленная почта

В листинге 19.1 приведен типичный файл `.fetchmailrc`.

Листинг 19.1. Образец файла `fetchmailrc`

```
set postmaster "ben"
set bouncemail
set syslog
set properties ""
set daemon 1800

poll franklin.threeroomco with proto IMAP4
    user "benny" there with password "be¥40u$" is ben here
    options fetchall forcecr smtphost osgood.threeroomco.com
poll pop.abigisp.net with proto POP3
    user "threeroom" there with password "u7 1qq4oO" is sally here
```

Этот файл состоит из двух частей.

- **Глобальные опции** В первых пяти строках листинга задаются глобальные параметры. Здесь указывается локальный пользователь, которому направляются сообщения об ошибках программы Fetchmail (`set postmaster`), а также определяются режим журнальной регистрации (`set syslog`) и частота проверки почты в режиме демона (`set daemon`)
- **Опрашиваемые серверы.** Каждая группа строк начинающаяся с ключевого слова `poll`, описывает сервер, с которого программа Fetchmail загружает почту. Строки с отступом являются продолжением предыдущей строки. В каждом случае нужно указывать всю информацию, необходимую для загрузки почты с сервера

Программа Fetchmail поддерживает огромное число опций. Все они подробно описаны на man-странице программы. В листинге 19.1 показана базовая конфигурация, которая подойдет для многих систем. Рассмотрим ее ключевые особенности.

- **Почтмейстер** (postmaster). Это локальный пользователь, просматривающий сообщения об ошибках и занимающийся прочими административными вопросами. Это не обязательно должен быть администратор электронной почты компьютера или домена, отвечающий за общее конфигурирование почтовой системы.
- **Режим демона** Как будет описано ниже, программа Fetchmail может работать в режиме демона, т.е. непрерывно. В этом случае параметр `set daemon` определяет интервал в секундах между операциями опроса серверов.
- **Порядок аргументов инструкции `poll`.** В инструкции `poll` сначала задается описание опрашиваемого сервера (его доменное имя, используемый протокол, имя пользователя и протокол), а затем указывается, что нужно делать с полученной почтой
- **Параметры размещения почты** В листинге 19.1 конфигурация сервера `pop.abigisp.net` довольно проста: вся полученная почта высылается пользователю `sally` локального компьютера. Что касается сервера `franklin.threeroomco.com`, то здесь задан ряд дополнительных параметров. Опция `smtphost` заставляет программу Fetchmail подключиться к компьютеру `osgood.threeroomco.com` для отправки полученной почты. Опция `fetchall` заставляет программу загружать сообщения даже в том случае, когда они дублируются. Опция `forcecr` меняет интерпретацию символа возврата каретки в сообщениях. Это используется при доставке почты серверу `qmail` (остальные серверы SMTP не "заботятся" о таком нюансе)

Большинство опций можно переопределять в строке вызова программы. Одна из опций командной строки заслуживает внимания, это --keep (или просто -k) Она заставляет программу сохранять сообщения на сервере (по умолчанию программа удаляет сообщения после загрузки) что очень удобно при тестировании. Запустив программу с этой опцией, можете быть уверены: в случае ошибки почта не потеряется

Чтобы не редактировать файл .fetchmailrc вручную, воспользуйтесь графической утилитой fetchmailconf. С ее помощью можно изучать опции программы. В любом случае конфигурационный файл должен быть доступен только пользователю, запускающему программу Fetchmail Это ограничение важно, поскольку файл содержит пароль доступа к удаленному серверу Разумеется, любые пароли представляют интерес для взломщиков

Запуск с помощью демона cron

Для одноразового запуска программы достаточно ввести ее имя в командной строке

```
$ fetchmail
```

Программа свяжется с удаленным сервером, загрузит имеющуюся почту и поместит ее в локальную почтовую очередь FreeBSD либо отправит ее на другой компьютер, если задана опция smtphost. Вызов программы разрешен пользователю в начальном каталоге которого находится файл .fetchmailrc.

Нет необходимости запускать программу от имени суперпользователя поскольку это лишь повышает риск ее применения. Если в коде программы есть ошибка, злоумышленник теоретически может заставить ее выполнить несанкционированные действия и тем самым получить контроль над компьютером.

Одноразовый запуск программы удобен в целях тестирования (особенно в сочетании с опцией --keep), но в большинстве случаев программа работает автоматически. Например, если есть коммутируемое PPP-соединение, можно написать сценарий, который будет устанавливать соединение и немедленно вызывать программу Fetchmail для загрузки почты Другой вариант — периодический запуск программы средствами демона cron. Это позволяет загружать почту на регулярной основе, скажем каждый час

В листинге 19.2 показан crontab-файл, организующий запуск программы за восемь минут до конца каждого часа с 6 52 утра до 18:52 вечера. Время загрузки смещено от начала часа, чтобы избежать перегрузки почтового сервера провайдера на тот случай, если другие пользователи тоже решат загружать почту каждый час. В этом примере результаты работы команды fetchmail перенаправляются в файл /dev/null, иначе пользователь указанный в переменной MAILTO crontab файла будет каждый час получать отчет о работе программы.

Листинг 19.2. Пример crontab-файла для запуска программы Fetchmail

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=ben
52 6-18 * * * /usr/local/bin/fetchmail > /dev/null
```

Если организуется периодический запуск программы, не используйте опцию set daemon. Когда она присутствует в конфигурационном файле, программа Fetchmail не завершает работу, оставаясь в фоновом режиме. Это не позволит демону cron запустить новый экземпляр программы

Запуск в режиме демона

Можно регулярно запускать программу Fetchmail с помощью демона `cron` или воспользоваться ее встроенными функциями демона. Для этого достаточно задать опцию `set daemon` в файле `.fetchmailrc` Программа начнет работать непрерывно, опрашивая почтовые серверы с периодичностью, указанной в опции `set daemon`. Период опроса задается в секундах. В листинге 19.1 периодичность составляет 1800 секунд, т.е. каждые полчаса

Прежде чем применять режим демона, нужно предусмотреть способ запуска и останова программы Напишите сценарий, вызывающий программу на этапе начальной загрузки системы. В случае однопользовательской рабочей станции можно включить команды запуска и останова в сценарии регистрации и выхода из системы.

К сожалению, в режиме демона программа Fetchmail не всегда работает надежно Если опрашиваемый сервер не отвечает, программа может прекратить прием почты По этой причине, а также из-за неудобств связанных с запуском и остановом программы, рекомендуется организовывать регулярный запуск программы средствами демона `cron` К тому же это обеспечивает большую гибкость поскольку, например, в листинге 19.2 программа загружает почту не каждый час, а лишь в те часы, когда ее поступление наиболее вероятно В режиме демона программа Fetchmail не сможет сделать паузу на ночь

Резюме

Обработка почты — важная функция многих FreeBSD-систем. Они могут функционировать в качестве почтовых серверов, обслуживающих множество пользователей и даже целые домены. Такие серверы используют протокол принудительной доставки почты SMTP, который реализован во множестве программ, включая `sendmail` — стандартный почтовый сервер FreeBSD В системе могут работать также серверы доставки почты по запросу, позволяющие почтовым клиентам Windows Mac OS FreeBSD и других операционных систем загружать почту полученную сервером SMTP Наконец, существует программа Fetchmail, которая умеет загружать почту с удаленного сервера POP или IMAP и автоматически помещать ее в почтовую очередь локальной или удаленной системы Это удобно в небольших сетях, в которых почта загружается с нескольких внешних серверов, а также в том случае когда в сети функционирует внутренний почтовый сервер.

Полный справочник по

Глава 20

Web-серверы

Для многих людей Internet отождествляется со всемирной паутиной (World Wide Web, WWW). Естественно, Internet — это не только WWW. В глобальном трафике присутствуют электронная почта, FTP, Telnet, SSH и другие протоколы. Однако WWW — самый заметный компонент глобальной сети. Язык HTML (Hypertext Markup Language), на котором пишутся Web-страницы, позволяет Web-броузерам формировать сложный, гибкий и визуально привлекательный интерфейс взаимодействия с пользователями.

Наиболее популярным Web-сервером в мире является Apache, который по оценкам компании Netcraft (`http://news.netcraft.com/archives/web_server_survey.html`) в 2003 г. обслуживал около 64% активных узлов. Apache в основном ориентирован на платформу UNIX и во FreeBSD работает достаточно хорошо. Вообще FreeBSD является отличной платформой для Web-сервера практически любого размера. Основным ограничивающим фактором является аппаратное обеспечение компьютера, например объем оперативной памяти и пропускная способность сети. В этой главе описывается конфигурирование Web-сервера Apache в среде FreeBSD. Будет также вкратце рассказано о принципах создания Web-страниц на языке HTML.

Когда нужно запускать Web-сервер

Первый вопрос, который необходимо задать себе перед инсталляцией Web-сервера, таков: имеет ли смысл создавать Web-узел самостоятельно? Web-узлы привлекают особое внимание взломщиков. В большинстве случаев попытки взлома являются лишь неприятной мелочью, но иногда ситуация оказывается действительно серьезной. Проблемы возникают как минимум в двух случаях:

- Если сервер уязвим к определенному виду атак, злоумышленник может получить доступ к компьютеру, чтобы повредить содержимое Web-узла, использовать систему в качестве стартовой площадки для нападения на другие системы, украсть данные или каким-то образом повлиять на работу системы.
- Действия злоумышленника могут быть направлены на перегрузку сетевого канала, что способно вывести Web-узел из строя. Атаки типа "отказ от обслуживания" наиболее эффективны в отношении небольших Web-узлов, подключенных к сетевым каналам с невысокой пропускной способностью.

В обоих случаях владелец Web-узла столкнется с неприятной необходимостью устранять последствия взлома, что может подразумевать переинсталляцию FreeBSD или многочасовые консультации с провайдером по поводу блокирования злоумышленника. Всего этого можно избежать, договорившись с провайдером о размещении Web-узла на его сервере. Тогда, если Web-узел подвергнется нападению, это станет проблемой провайдера. К тому же владельцу Web-узла придется гораздо меньше заниматься вопросами конфигурирования Web-сервера.

Но и самостоятельный запуск Web-сервера имеет свои преимущества. Во-первых, вы будете иметь полный контроль над сервером. Многие провайдеры, предоставляющие услуги Web-хостинга, ограничивают возможности владельцев Web-узлов, начиная от объема доступного дискового пространства и заканчивая типами выполняемых Web-сценариев. В случае самостоятельного контроля вы сможете конфигурировать сервер по своему усмотрению, а также организовать его взаимодействие с другими системами, например с внутренними базами данных. Второе преимущество заключается в том, что сервер можно сконфигурировать исключительно для внутреннего использования.

Web-серверы для FreeBSD

Apache — наиболее популярный Web-сервер для FreeBSD но далеко не единственный. Существует целый ряд других аналогичных программ

- **Apache** (`http://httpd.apache.org`) Самый распространенный Web-сервер для UNIX-систем вообще и FreeBSD в частности Он обладает богатыми функциональными возможностями, и его конфигурация может быть довольно сложной, однако стандартная конфигурация вполне подходит для небольших Web-узлов, что делает этот Web-сервер одним из наиболее простых в использо
- **Roxen** (`www roxen.com`). Конкурент Apache в плане функциональных возможностей и общей производительности Сервер предназначен для крупных Web узлов со сложной конфигурацией
- **thttpd** (`http://www.acme com/software/thttpd/`) Небольшой (50 Кбайт против 330 Кбайт у Apache) и высокопроизводительный сервер Размеры сервера позволяют ему обслуживать интенсивно посещаемые Web-узлы на относительно "слабом" оборудовании.
- **Mathopd** (`www.mathopd.org`). Как и `thttpd` это небольшой и производительный сервер, ориентированный на обработку большого числа одновременных соединений
- **Boa** (`www.boa.org`). Еще один небольшой Web-сервер. Он особенно популярен во встроенных системах, т е в компьютерах которые встраиваются в различные устройства такие как газовые насосы и бытовая техника
- **Специализированные средства** Некоторые программы обычно не рассматривают как Web-серверы но они взаимодействуют с зарезервированными для них портами по протоколу HTTP, что позволяет использовать их с обычными Web-броузерами а не писать специализированные клиенты В качестве примеров укажем программу SWAT (Samba Web Administration Tool), которая реализует графический интерфейс для администрирования пакета Samba и системную административную утилиту Webmin (`www.webmin.com`).

Оптимальным выбором для FreeBSD является сервер Apache, который поставляется вместе с системой Именно он будет описан в этой главе. Серверы `thttpd`, Mathopd и Boa доступны в виде пакетов или переносимых приложений и хорошо подходят для простых Web узлов даже если через них проходит интенсивный трафик.

Основы конфигурирования сервера Apache

Несмотря на сложность сервера Apache, запустить его в базовой конфигурации совсем не трудно, поскольку при инсталляции сервера во FreeBSD создаются вполне функциональные конфигурационные файлы После запуска сервер будет готов обслуживать HTML-файлы (о создании таких файлов рассказывается в следующем разделе). Конфигурационные изменения могут касаться способа запуска сервера и каталогов, используемых для хранения файлов. Нужно также знать, как загружаются модули Apache, которые реализуют расширенные функциональные возможности сервера

На заметку В этой главе описывается Apache 2.0. Серверы версии 1.3.*x* до сих пор используются и тоже доступны во FreeBSD в виде пакетов и переносимых приложений

Запуск Apache

Как указывалось в главе 16, многие серверы можно запускать либо автономно, ли бо под управлением супердемона inetd. Это было справедливо для Apache 1.3.x. В конфигурационном файле присутствовал параметр ServerType, определявший способ запуска сервера standalone или inetd. Но уже тогда разработчики сервера не реко мендовали использовать его совместно с демоном inetd: это ухудшало производи тельность сервера, особенно в случае интенсивного трафика, и не способствовало по вышению надежности.

Сервер версии 2 работает только автономно. В пакет Apache входит сценарий /usr/local/etc/rc.d/apache.sh, который запускает сервер на этапе начальной загрузки системы. Чтобы проверить, работает ли сервер, воспользуйтесь командой ps и поищите процесс httpd:

```
$ ps ax | grep httpd
  456  ??  Ss     0:00.12 /usr/local/sbin/httpd
  457  ??  I      0:00.01 /usr/local/sbin/httpd
  458  ??  I      0:00.03 /usr/local/sbin/httpd
  459  ??  I      0:00.02 /usr/local/sbin/httpd
  460  ??  I      0:00.02 /usr/local/sbin/httpd
  461  ??  I      0:00.02 /usr/local/sbin/httpd
  462  ??  I      0:00.03 /usr/local/sbin/httpd
  489  p1  R+     0:00.01 grep httpd
```

Как показано в этом примере, сервер Apache создает несколько экземпляров са мого себя. Это позволяет серверу при поступлении запроса на подключение быстро перенаправить его свободному процессу. Каждый процесс обрабатывает один запрос. Если число клиентов превышает количество процессов Apache, сервер порождает до полнительные копии самого себя.

Чтобы запустить сервер непосредственно после инсталляции, выполните следую щую команду:

```
# /usr/local/sbin/httpd
```

На заметку Сервер не нужно перезапускать при изменении файлов на Web-узле, но это нужно делать после модификации конфигурационного файла. Соответствующая команда такова: **killall -HUP httpd**.

Конфигурационный файл

Конфигурационный файл сервера Apache 2 называется /usr/local/etc/apache2/httpd.conf. Установки этого файла называются директивами. Их формат таков.

```
Директива значение
```

Некоторые директивы определяют группу значений. Они берутся в угловые скобки (<>), а в конце группы в угловых скобках ставится косая черта и название директивы. Промежуточные строки содержат другие директивы. К примеру, в стандартном кон фигурационном файле встречаются следующие установки:

```
<Directory "/usr/local/www/data">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

Они задают параметры обработки конкретного каталога сервера. Подробнее об этом будет рассказываться на с. 397.

Символ '# обозначает комментарий. Строки, начинающиеся с этого символа игнорируются. Стандартный файл `httpd.conf` содержит множество комментариев, из которых можно многое узнать о структуре файла и назначении директив

Базовые параметры сервера

В простейшей конфигурации большинство директив стандартного файла `httpd conf` можно оставить без изменений Ниже перечислены опции которые чаще остальных требуют модификации.

- **Параметры запуска резервных серверов.** Директивы `StartServers`, `MinSpareServers` и `MaxSpareServers` определяют, сколько процессов Apache должно быть запущено. Директива `StartServers` задает начальное число серверов Директивы `MinSpareServers` и `MaxSpareServers` указывают, какое количество резервных серверов (тех, что не занимаются обработкой запросов) может выполняться. Когда число запросов возрастает, Apache запускает дополнительные экземпляры самого себя, чтобы ускорить обработку запросов. При этом часть процессов (`MinSpareServers`) постоянно остается свободной, чтобы справляться с пиковыми всплесками трафика По мере снижения нагрузки резервные процессы уничтожаются, пока их число не станет меньше значения `MaxSpareServers` Для небольших Web-узлов вполне подойдут стандартные значения 5, 5 и 10 Значение `MaxSpareServers` должно быть больше, чем `MinSpareServers` Все эти директивы объединяются директивой `IfModule`. Во FreeBSD используется набор директив `<IfModule prefork.c>` Остальные аналогичные наборы предназначены для запуска Apache на других платформах.
- **`MaxClients`** — задает максимальное число клиентов, обслуживаемых сервером одновременно Значение по умолчанию равное 150, подходит для небольших и среднего размера Web-узлов Эта директива играет важную роль в защите сервера от атак типа "отказ от обслуживания", когда сервер становится недоступным из-за слишком большого числа поступивших запросов.
- **`Listen`** — определяет порт, к которому подключается сервер Apache (по умолчанию 80, т.е. традиционный порт протокола HTTP) Нестандартные порты задаются, когда на одном компьютере работает несколько серверов. Дополнительно можно указать IP-адрес, чтобы сервер Apache подключался к нужному порту только одного сетевого интерфейса, например `172.17.2.217:80`
- **`User` и `Group`**. Эти директивы сообщают серверу, какие учетные записи нужно использовать для запуска различных подпроцессов Стандартное значение `www` подходит в большинстве случаев, но можно создать новую учетную запись.

Внимание! Не запускайте сервер от имени пользователя `root` и даже от имени рядового пользователя с обычными привилегиями. Это существенно повышает риск взлома сервера

- **`ServerAdmin`** — задает почтовый адрес администратора Web-сервера. Указанное значение сообщается клиентам при возникновении определенных ошибок
- **`ServerName`** — задает имя и порт сервера, например `www.threeroomco:80`. Apache автоматически определяет корректные значения, но не помешает подстраховаться на тот случай если в момент запуска Apache DNS-сервер окажется недоступен.
- **`TypesConfig`** Файлы, передаваемые по протоколу HTTP, содержат код MIME (Multipurpose Internet Mail Extension — многоцелевые расширения электронной почты в сети Internet), идентифицирующий тип файла (обычный текст графи-

ческое изображение и т.д.). Директива `TypeConfig` задает файл, в котором хранится таблица соответствий между расширениями файлов и кодами MIME. В этот файл можно добавлять описания новых типов данных.

- **`DefaultType`** Если Apache не может определить тип файла, используется тип, заданный в директиве `DefaultType`. По умолчанию это `text/plain`, т.е. обычный текст.
- **`HostnameLookups`** Эта директива может иметь значение `On` или `Off` и определяет, должен ли Apache регистрировать доменные имена клиентов или только их IP-адреса. Регистрация доменных имен — полезная возможность, но она повышает нагрузку на сервер, поскольку для каждого соединения приходится выполнять поиск в DNS.

Это далеко не полный перечень директив, встречающихся в файле `httpd.conf`. Единственная из них, которую точно придется изменить, — `ServerAdmin`. Остальные можно модифицировать для повышения производительности и безопасности, а также для того, чтобы сервер лучше взаимодействовал с клиентами.

Существует ряд директив, связанных с доставкой Web-страниц. В основном они касаются определения каталогов, используемых для хранения Web-страниц. Об этом пойдет речь далее.

Параметры серверных каталогов

Чтобы загрузить нужную Web-страницу, пользователь вводит в Web-броузере ее URL-адрес, который выглядит примерно так:

```
http //www threeroomco.com/products/fb50 h ml
```

На заметку Это типичный пример URL-адреса простой Web-страницы. В адресе могут присутствовать дополнительные компоненты.

URL-адрес содержит информацию об используемом протоколе (`http`), компьютере, с которым осуществляется связь (`www.threeroomco.com`), и файле, загружаемом с этого компьютера (`products/fb50 html`). Последний компонент задает каталог и файл. Нежелательно предоставлять клиентам доступ ко всем файлам компьютера. Обычно определяется один или несколько каталогов, хранящих Web-страницы, и все адреса вводятся относительно этих каталогов. В показанном примере `products` — подкаталог корневого каталога документов Web-сервера, а не корневого каталога системы. Эти и другие установки, касающиеся обработки запросов к каталогам, задаются с помощью специальных директив.

- **`DocumentRoot`** определяет корневой каталог для хранения документов. Значение по умолчанию — `"/usr/local/www/data"` (кавычки необходимы).
- **`UserDir`** Web-сервер может хранить файлы (персональные Web-страницы) в каталогах рядовых пользователей. Директива `UserDir` задает подкаталог начального каталога пользователя, в котором следует искать такие файлы. Например, если файл `httpd.conf` сервера `www.threeroomco.com` содержит директиву `UserDir public_html` (это установка по умолчанию) и клиент запрашивает документ `http·//www.threeroomco.com/~amy/mypage.html`, сервер Apache возвращает файл `public_html/mypage.html` из начального каталога пользователя `amy` (полное имя файла, очевидно, `/home/amy/public html/mypage.html`). Тильда (~) в URL-адресе означает, что файл находится в начальном каталоге пользователя.

- **Directory** Эта директива начинает блок установок, поэтому берется в угловые скобки Ее значением является имя каталога (заданное относительно корневого каталога системы), к которому относятся установки например <Directory /> или <Directory /usr/local/www/data>. Оба указаниых блока встречаются в стандартном конфигурационном файле Первый из них ограничивает доступ к пользовательским каталогам (которые могут находиться где угодно), второй задает параметры доступа к основному каталогу сервера. Второе значение должно соответствовать директиве DocumentRoot. Блок установок каталога заканчивается строкой </Directory> В нем могут переопределяться глобальные установки
- **AllowOverride** Apache позволяет пользователям создавать файлы, содержащие директивы сервера Эти файлы называются .htaccess. Директива AllowOverride определяет, должен ли сервер Apache читать их и применять соответствующие директивы. Значения All и None, соответственно, разрешают и запрещают все директивы. Дополнительные опции описаны в документации к серверу. Значение по умолчанию None является оптимальным для простого Web-узла. Эта директива обычно встречается в блоке директив <Directory>
- **DirectoryIndex**. Если в URL-адресе не указано имя файла, Apache возвращает файл, имя которого определено в директиве DirectoryIndex. Как правило значение по умолчанию равно index html Эта директива обычно встречается в блоке директив <Directory>.

Модули Apache

Сервер Apache, как и ядро FreeBSD, можно компилировать модульным способом В этом случае основной серверный файл остается небольшим, а дополнительные файлы загружаются в виде модулей по мере необходимости, что существенно снижает требования к оперативной памяти, особенно для относительно простых Web-узлов. Повышается также безопасность сервера, поскольку потенциально опасный код большую часть времени недоступен Поддержка модулей позволяет разработчикам легко расширять функциональные возможности Apache, не перекомпилируя весь сервер. Разработчику достаточно написать модуль взаимодействующий с ядром Apache посредством четко определенных методов

В пакет Apache входит много стандартных модулей. Дополнительные модули можно найти в Internet. В частности, на Web-узле Apache Module Registry (http://modules apache.org) приведены ссылки на различные проекты, связанные с разработкой модулей Apache, и есть механизм поиска модулей.

С обработкой модулей связано множество директив Ниже описаны наиболее важные из них

- **LoadModule** — загружает модуль из файла на диске В качестве аргументов указываются имя модуля и имя его файла (как правило, приводится полный путь к файлу). В стандартном файле httpd.conf содержится большое число директив LoadModule, загружающих стандартные модули.
- **AddModule** — активизирует модуль, встроенный в исполняемый код Apache или загруженный из файла. Обычно в этом нет необходимости и в файле httpd.conf данная директива не используется
- **IfModule** — начинает блок установок, применяющихся только в том случае, если указанный модуль присутствует, например <IfModule prefork c>

(модуль `prefork.c` встроен в исполняемый код сервера Apache во FreeBSD) Смысл директивы можно изменить на противоположный, если поставить перед именем модуля оператор '!'. Так, директива `<IfModule !mpm_netware.c>` обозначает блок директив, применяющихся только тогда, когда модуль `mpm_netware.c` отсутствует.

Для получения списка модулей, встроенных в исполняемый код Apache вызовите демон `httpd` с опцией `-l`

```
$ /usr/local/sbin/httpd -l
Compiled in modules:
  core.c
  prefork.c
  http_core.c
  mod_so.c
```

Остальные модули загружаются директивами `LoadModule` в файле `httpd.conf` Чтобы подключить дополнительный модуль, инсталлируйте его (при необходимости) и добавьте соответствующую директиву `LoadModule`.

В простейшей конфигурации серверу Apache вряд ли нужны дополнительные модули помимо тех, что включены изначально Чаще приходится отключать стандартные модули Например, по умолчанию в файле `httpd.conf` присутствует такая строка

```
LoadModule cgi_module libexec/apache/mod_cgi.so
```

Она включает поддержку CGI (Common Gateway Interface — единый шлюзовой интерфейс) CGI позволяет серверу выполнять сценарии, динамически генерирующие Web-страницы Это удобно в случае Web-узлов, вид которых изменяется при обработке запросов (например, поисковые системы и электронные магазины). Но в то же время возникает угроза безопасности, поскольку плохо написанный CGI-сценарий может стать "черным ходом" для взломщика Если Web-сервер не занимается обработкой CGI сценариев, превратите показанную строку в комментарий Другие модули тоже могут оказаться ненужными В частности, модуль `userdir_module` отвечает за обработку пользовательских каталогов. Если таковых нет отключите модуль

Создание Web-страниц

Запустить Web-сервер Apache несложно, поскольку его стандартная конфигурация вполне походит для простого Web-узла. Но это лишь часть задачи Реальному Web узлу нужно содержимое — набор текстовых документов в формате HTML. Сложные Web-узлы сдержат сотни и даже тысячи HTML-файлов.

Структура HTML-документов очень проста, их можно создавать в обычном текстовом редакторе. Существуют также специализированные HTML-редакторы, автоматизирующие создание таблиц, графики и прочих элементов интерфейса, код для которых слишком утомительно писать вручную. При написании HTML-документов следует помнить о том что Web-броузеры могут по-разному интерпретировать один и тот же HTML-файл, причем расхождения бывают весьма существенными Наконец CGI сценарии позволяют динамически менять вид Web-страницы для каждого пользователя, но это достигается за счет возрастания сложности кода и снижения безопасности

На заметку

Стандартный HTML-документ воспринимается любым Web-сервером будь-то Apache, `thttpd` или IIS (Internet Information Server) для Windows. Некоторые расширения, например CGI-сценарии, доступны не для всех серверов. Кроме того, серверы могут по-разному называть индексный файл (в Apache он задается директивой `DirectoryIndex`)

Структура HTML-документа

HTML-документы являются обычными текстовыми файлами в которых определенные выражения интерпретируются по-особому В частности, угловые скобки (<>) обозначают *HTML-теги* — специальные коды, указывающие на то, где начинается и заканчивается специальное форматирование текста. Символ амперсанда ('&'), стоящий перед ключевым словом, используется для вставки в документ специальных символов, например самих угловых скобок В листинге 20.1 показан пример HTML-документа

Листинг 20.1. Образец HTML-файла

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 4 01//EN >
<HTML><HEAD>
<TITLE>HTML Sample</TITLE>
</HEAD>
<BODE BGCOLOR="#FFFFFF" TEXT="#000000">
<CENTER><H1 ALIGN="CENTER">Sample Web Page</H1></CENTER>

<P>This is a <strong>sample</strong> web page. Read the
<A HREF="http://www.w3 org/TR/html401/">HTML definition</a>
for more <EM>great</EM> information!</P>

<P><IMG SRC="logo.jpg" ALT "Our Logo" ALIGH  LEFT  WIDTH= 200"
HEIGHT="150"></P>

<BR CLEAR=ALL>

<P>Copyright &copy  2002 </P>

</BODY>
</HTML>
```

Рассмотрим основные компоненты этого файла

- **Тип документа** Тег <!DOCTYPE> начинает большинстао HTML файлов. Он определяет версию стандарта HTML, которому соответствует файл (в нашем случае — 4 01) Впрочем, броузеры смогут прочесть файл и без этого тега
- **Тег <HTML>** — помечает начало HTML кода В самом конце файла его дополняет "закрывающий" тег </HTML>
- **Тег <HEAD>** HTML-файлы обычно состоят из двух частей *заголовка*, который содержит служебную информацию такую как строка заголовка окна, и *тела*, состоящего из отображаемого текста Заголовок заключается в теги <HEAD> и </HEAD>
- **Тег <TITLE>** — совместно с закрывающим тегом </TITLE> помечает строку отображаемую в заголовке окна Web-броузера
- **Тег <BODY>** — помечает начало тела документа которое заканчивается тегом </BODY> в конце файла. Тег <BODY> поддерживает дополнительные параметры, задающие цвет фона (BGCOLOR="#FFFFFF", т.е. белый) и цвет текста (TEXT="#000000", т.е. черный) Если используется фоновое изображение оно тоже задается в теге <BODY> (BACKGROUND="*имя_файла.расширение*").
- **Тег <CENTER>** — заставляет броузер центрировать текст, завершающийся тегом </CENTER>. Аналогичные теги <RIGHT> и <LEFT> включают выравнивание по правому и левому краю Они не используются большинством современных броузеров, поскольку соответствующие параметры есть в других тегах В листинг 20 1 включен тег <CENTER>, поскольку он, во-первых, не причиняет никакого вреда, а во-вторых, повышает переносимость документа

- **Теги заголовков** Теги типа <H1> обозначают заголовки текста, наподобие тех которые есть в этой книге В листинге 20 1 тег <H1> содержит параметр ALIGH="CENTER" Это более современный способ выравнивания текста. Под держиваются текстовые заголовки более низкого уровня. <H2>, <H3> и т.д.
- **Теги абзацев.** Теги <P> и </P> помечают начало и конец абзаца. Перенос слов в пределах абзаца определяется самим броузером. Если нужно начать текст с новой строки, заключите его в указанные теги или воспользуйтесь тегом
 (описан ниже)
- **Гиперссылки** создаются с помощью тега <A> В параметре HREF указывается URL адрес документа Конец выделяемого текста помечается тегом </A> Ко гда пользователь щелкает на гиперссылке, броузер пытается обратиться по ука занному адресу. В качестве объекта ссылки может выступать другая Web-страница (http://) файл на FTP-узле (ftp.//) или почтовый адрес (mailto:)
- **Выделенный текст** Теги <STRONG> и <EM> реализуют два способа выделения текста Большинство броузеров делает текст полужирным в первом случае и курсивным во втором, но могут быть и исключения.
- **Графические файлы.** Тег <IMG> задает графический файл, отображаемый Web-броузером В листинге 20 1 использовано несколько параметров этого тега, наиболее важный из которых — SRC, задающий источник получения файла Это может быть собственно имя файла (возможно, с указанием пути к нему) в таком случае клиент загрузит изображение с того же сервера, с которого была получена основная страница; или полный URL-адрес, указывающий на совер шенно другой сервер Параметр ALT задает короткое текстовое описание изо бражения Оно может быть важно для пользователей, которые сконфигуриро вали броузеры так, чтобы изображения не загружались автоматически. Пара метры WIDTH и HEIGHT задают размеры изображения в пикселях, что позволяет броузерам корректно масштабировать его
- **Тег
** задает разрыв строки Параметр CLEAR=ALL указывает на необхо димость удаления всех промежуточных элементов В результате последующий текст появится ниже изображения. Без тега
 некоторые броузеры отобра зят текст справа от изображения
- **Специальные символы** В листинге 20 1 используется только один специальный символ — знак авторского права обозначаемый как © Используются также специальные выражения < и &rt;, обозначающие символы "меньше" (<) и "больше" (>) Эти символы нельзя вводить непосредственно поскольку они интерпретируются как ограничители тегов.

Допускается вложение тегов. Например можно создать текст выделенный одно временно полужирным и курсивом

```
<STRONG><EM>A very emphasized point'</EM></STRONG>
```

Убедитесь в том, что вложенные теги закрываются в обратном порядке В данном примере теги <STRONG> и </STRONG> охватывают всю строку, включая теги <EM> и </EM>.

HTML - очень богатый язык, поддерживающий множество расширенных средств оформления Web-страниц С его помощью можно создавать таблицы, списки, формы и многое другое. Полная спецификация стандарта HTML 4 01 приведена по адресу http://www.w3.org/TR/html401/. Учебные материалы по языку HTML можно най ти на следующих Web-узлах.

```
http://www.davesite.com/webstation html/
http://www.w3schools.com/html/
http://www.wdvl.com/Authoring/HTML/
http://hotwired.lycos.com/webmonkey/
```

Изучайте также существующие HTML-документы. Большинство броузеров содер жит команду позволяющую просмотреть Web-страницу в формате HTML и сохранить ее для последующего использования. К сожалению, современные Web-узлы часто создаются с помощью программ, которые автоматически генерируют HTML-код трудный для понимания Тем не менее анализ Web-страниц позволяет получить много полезной информации особенно если нужно разобраться в том как реализуется тот или иной прием

Средства создания Web-страниц

Web-страницы можно писать на языке HTML вручную если вы знаете пару десят ков основных тегов. Но многие пользователи предпочитают применять графические средства разработки Web-страниц. В действительности существует несколько классов средств разработки

- **Обычные текстовые редакторы**. Любой текстовый редактор позволяет писать HTML-код вручную. Некоторые из них содержат средства, упрощающие эту задачу. Например, может поддерживаться *синтаксическое выделение*, когда ключевые структурные элементы, например теги, помечаются особым цветом.
- **Текстовые процессоры**. Многие современные текстовые процессоры способны экспортировать файлы в формате HTML. Впрочем, далеко не все возможности таких программ поддерживаются самим языком HTML, поэтому расширенное форматирование может теряться Текстовые процессоры позволяют преобразовывать существующие документы в формат HTML, однако генерируемый ими HTML-код получается хуже, чем у специализированных HTML-редакторов.
- **Web-броузеры**. Некоторые Web-броузеры, например Netscape содержат специа лизированные средства разработки Web-страниц. Они меньше подвержены проблемам, связанным с импортом/экспортом, чем текстовые процессоры.
- **HTML-конвертеры**. Существуют программы, предназначенные для преобразо вания файлов в формат HTML К примеру, программа `c2html` преобразует ис ходный код на языке C, а программа `cthumb` создает Web страницу в виде фо тоальбома
- **Специализированные HTML-редакторы**. Существует также немало программ, предназначенных специально для редактирования HTML-файлов Большинство из них, в частности Bluefish, `erwin`, Gnotepad+, Peacock, Quanta и Scream, в сущности являются обычными текстовыми редакторами, отображающими HTML-код в режиме синтаксических выделений и содержащими различные команды меню для вставки HTML-тегов и прочих элементов Некоторые про граммы имеют режим предварительного просмотра в котором HTML документ отображается так, как он будет выглядеть в Web-броузере Редакторы наподобие ASHE (`ftp://ftp.cs.rpi.edu/pub/puninj/ASHE/README html`) работают по принципу WYSIWYG (What You See Is What You Get — что видишь, то и получишь), отображая форматированный текст в точностн так как и большинство броузеров.

Используя графические средства разработки помните что режимы предваритель ного просмотра не соответствуют принципу WYSIWYG в той степени, какая свойственна текстовым процессорам при отображении текста Web-броузеры могут выполнять (и выполняют) переформатирование страниц чтобы они соответствовали пара

метрам клиентской системы. Шрифты, разрывы строк, цвета, цветовая разрядность и многие другие элементы меняются от системы к системе. По этой причине следует тестировать Web-страницу на как можно большем числе платформ.

Работают с HTML-редактором точно так же, как с любым текстовым редактором Просто в нем есть команды меню и панели, ориентированные на создание Web-страниц. К примеру, на рис. 20.1 приведено окно редактора Bluefish Помимо основной панели редактирования и стандартной панели инструментов, в нем есть меню и кнопки, специфичные для Web-дизайна, например меню Tags

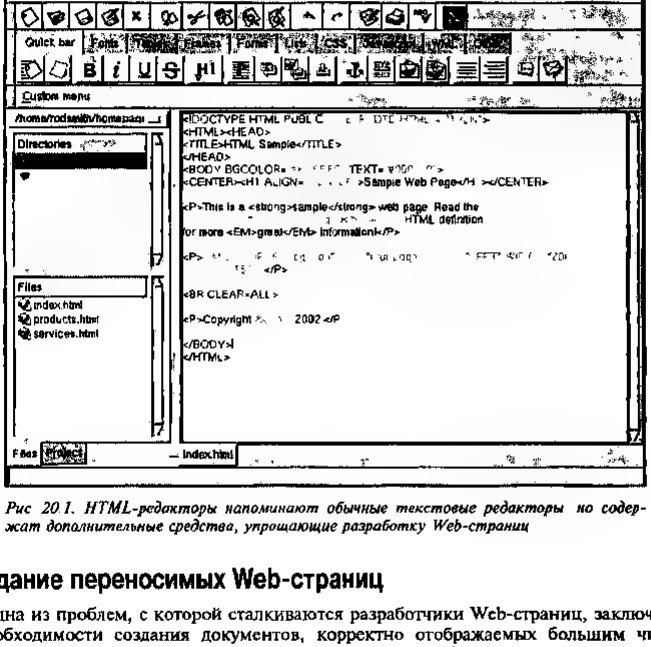


Рис 20 1. HTML-редакторы напоминают обычные текстовые редакторы но содержат дополнительные средства, упрощающие разработку Web-страниц

Создание переносимых Web-страниц

Одна из проблем, с которой сталкиваются разработчики Web-страниц, заключается в необходимости создания документов, корректно отображаемых большим числом Web-броузеров Даже Internet Explorer компании Microsoft, наиболее широко используемый сегодня, бывает разных версий, у каждой из которых уникальные характеристики Немалую долю рынка занимают Netscape Navigator, Mozilla, Opera, Konqueror, `lynx` и другие броузеры. Все они интерпретируют Web-страницы по-разному

Более того, сами пользователи конфигурируют свои Web-броузеры неодинаково Они могут менять стандартные цвета, стандартные шрифты, стандартные размеры шрифтов, размеры окон, правила вывода изображений и т.д. Некоторые возможности в ряде броузеров вообще отсутствуют К примеру `lynx` – это текстовый Web броузер поэтому он не может отображать графику

Язык HTML задумывался как межплатформенное средство, однако HTML редакторы, особенно редакторы WYSIWYG и текстовые процессоры, часто генерируют недостаточно переносимый HTML-код Создаваемые ими документы могут содержать ошибки (например, теги в неправильном порядке), которые нарушают работу броузеров. Другого рода проблемы возникают вследствие ошибок проектирования когда делается слишком много предположений о конфигурации броузера Разработчик может задать цвет фона, но не позаботиться об установке цвета текста. И если пользователь выберет в броузере цвет текста, близкий к цвету фона страницы, текст станет нечитаемым Вот почему так важно тестировать Web-страницы

- **Проверка синтаксиса** Существуют программы, проверяющие Web страницы на совместимость со стандартами языка HTML Во FreeBSD это умеет делать текстовая утилита `weblint` (доступна в виде пакета и переносимого приложения) Введите **`weblint`** *`имя_файла`*`.html`, чтобы получить отчет о соответствии заданного файла базовым стандартам. Некоторые HTML-редакторы вызывают эту утилиту самостоятельно или же содержат собственные средства проверки
- **Проверка понятности** Существуют программы и службы, которые находят на Web-страницах типичные ошибки, снижающие их понятность. Наиболее популярная из этих программ называется Bobby (`http://bobby.watchfire.com/bobby/`) Она способна проверить любую Web-страницу в Internet Программа сообщает о потенциальных проблемах, таких как "наплывание" гиперссылок, отсутствие параметра `ALT` у тега `<IMG>` и т.д.
- **Проверка вручную в различных броузерах** Нет ничего надежнее, чем самостоятельно загрузить Web-страницу в различные броузеры и посмотреть результат. Обязательно включите в проверку одну из последних версий Internet Explorer, так как это наиболее популярный броузер Рекомендуется также выполнять тестирование в броузере `lynx`, который популярен в системах с текстовыми мониторами С его помощью можно создавать Web узлы, доступные людям с нарушенным зрением, которые для просмотра Web-страниц используют синтезатор речи. Проверяйте Web-страницы при разном разрешении экрана и разной цветовой разрядности

Выполнив такие проверки, можно быть уверенным в том, что Web страница будет нормально отображаться в большинстве случаев Естественно, стопроцентной гарантии не существует. Код броузера может содержать ошибку, которая вызывает сбой при получении определенной HTML-последовательности, даже если эта последовательность совершенно легальна и приемлема для других броузеров. Но такие случаи все же редки, да и программный код броузеров постоянно совершенствуется

Резюме

FreeBSD является прекрасной платформой для Web-серверов, поскольку это стабильная операционная система, поддерживающая большое число серверных программ, включая надежный и популярный сервер Apache. Настроить Apache нетрудно, так как стандартный конфигурационный файл вполне подходит для простого Web-узла В более сложных конфигурациях может потребоваться модифицировать различные параметры, касающиеся каталогов Web-страниц, модулей и других компонентов.

После того как Web-сервер запущен, нужно подготовить файлы, которые сервер будет передавать клиентам Такие файлы в основном пишутся на языке HTML. Создавать HTML-страницы можно с помощью обычного текстового редактора или специализированного HTML-редактора

Полный справочник по FreeBSD

Глава 21

Серверы дистанционной регистрации

Многие серверные программы выполняются на компьютерах которые предос тавляют пользователям лишь самый ограниченный доступ Например, пользо ватель, просматривающий документы на Web-сервере, обычно даже не располагает учетной записью на нем Пользователи серверов доставки почты по запросу имеют учетные записи, но с привилегиями, допускающими только загрузку почты В этой главе рассматривается класс серверов, обеспечивающих гораздо более широкий доступ к компьютеру: *серверы дистанционной регистрации*. Они разрешают пользователям регистрироваться в системе и запускать любые программы Такие серверы могут рабо тать в крупных многопользовательских системах, избавляя от необходимости задейст вовать большое число рабочих станций Пользователи смогут одновременно получать доступ к одному компьютеру с обычных терминалов. Другое распространенное при менение серверов дистанционной регистрации — предоставление пользователям уда ленного доступа к своим рабочим станциям

Эта глава начинается с краткого описания типов серверов дистанционной регист рации после чего рассматриваются примеры конкретных серверов. В качестве серве ров, работающих в текстовом режиме, выбраны Telnet и SSH, а в качестве серверов с графическим интерфейсом — X Window System и VNC

Типы серверов дистанционной регистрации

В 70-е годы, когда операционная система UNIX только начинала развиваться, ра бота за компьютером в основном велась в текстовом режиме Многие средства тексто вого режима до сих пор сохранились в UNIX, в том числе и во FreeBSD поэтому протоколы дистанционного доступа, ориентированные на работу с обычными тексто выми данными, могут оказаться очень удобными для работы в системе на расстоянии Вы сидите за своим компьютером, вводите команды и наблюдаете результаты их вы полнения на другом компьютере. В этом суть серверов дистанционной регистрации работающих в текстовом режиме: благодаря им FreeBSD воспринимает удаленный компьютер как локальную текстовую консоль а удаленные пользователи могут вызы вать программы FreeBSD.

Существует много протоколов регистрации в текстовом режиме, из которых в этой главе будут рассмотрены два Telnet и SSH (Secure Shell — защищенный интерпрета тор команд). Первый из них является более старым и простым. Он реализован во всех FreeBSD-системах. Клиенты Telnet есть в большинстве операционных систем, вклю чая все разновидности UNIX, Windows, OS/2 и Mac OS X (в более ранних версиях Mac OS в качестве клиентов Telnet использовалось внешнее программное обеспече ние) Несмотря на распространенность, Telnet — далеко не лучший протокол дистан ционной регистрации, поскольку все данные в нем передаются в незашифрованном виде Перехватив поток данных злоумышленник получает доступ к конфиденциаль ной информации, передаваемой в рамках сеанса, в частности к паролям. Встречаются разновидности протокола, поддерживающие шифрование, например реализация Telnet в Kerberos (`http://web.mit.edu/kerberos/www/`), но они не приобрели той популярности, которую имеет исходный вариант Telnet.

SSH — это шифрованный протокол дистанционной регистрации в текстовом ре жиме, обладающий целым рядом расширенных функциональных возможностей В SSH реализованы *туннели*, т.е в рамках SSH-соединения клиент и сервер могут пере давать данные других протоколов. Имеются также встроенные средства передачи фай лов. Все это делает SSH чрезвычайно мощным протоколом, гораздо более безопас ным, чем Telnet В качестве недостатка можно указать то, что для шифрования требу ются ресурсы процессора, а это приводит к снижению производительности особенно при туннелировании данных высокоскоростных протоколов

С появлением операционных систем с графическим интерфейсом протоколы доступа в текстовом режиме перестали удовлетворять определенным потребностям пользователей. Например, с их помощью нельзя запустить графический редактор, такой как GIMP. Для решения этих задач было разработано новое семейство протоколов. Один из членов этого семейства — встроенная графическая среда FreeBSD· X Window System (сокращенно — X-среда) В отличие от графических оболочек других операционных систем она изначально проектировалась с расчетом на работу в сети, поэтому ее легко использовать для удаленного доступа, по крайней мере теоретически. На практике проблемы безопасности и множество параметров, определяющих способы установки соединения, делают удаленный доступ в X-среде более сложным, чем в случае Telnet или SSH

Более современным графическим средством удаленного доступа является VNC (Virtual Network Computing — виртуальная среда сетевых вычислений; `www.realvnc.com`) Она ориентирована на удаленный доступ к различным графическим средам, включая X Window System, Windows и Mac OS Во FreeBSD VNC применяется следующим образом в системе, где работают программы, X Window System выполняется в локальном режиме, а VNC пересылает отображаемые данные в удаленную систему В определенных ситуациях VNC проще конфигурировать, чем встроенные средства удаленного доступа X Window System, однако эта кажущаяся простота достигается за счет дополнительного уровня программных средств Во многих сетях VNC работает медленнее, чем X Window System

Во FreeBSD поддерживаются все вышеперечисленные средства удаленного доступа В текстовом режиме SSH будет оптимальным выбором. В графическом режиме у X Window System и VNC есть как преимущества, так и недостатки. Как правило, если система, в которой работает пользователь, относится к семейству UNIX, то лучше применять X Window System напрямую а в Windows и Mac OS проще конфигурировать клиенты VNC

Конфигурирование сервера Telnet

Telnet — один из простейших и популярнейших протоколов удаленного доступа в текстовом режиме. В то же время он печально известен своей незащищенностью Данные в нем не шифруются, так что любой, кто сумеет перехватить пакеты на стороне отправителя или получателя либо на промежуточной системе, сможет узнать пароль пользователя. По этой причине рекомендуется вместо Telnet использовать более безопасный протокол SSH.

Однако в некоторых случаях выбор Telnet неизбежен. Например, не всегда можно быть уверенным в наличии у пользователей клиентов SSH. Риск, связанный с применением Telnet, приемлем также в небольших частных сетях. Ниже приводится информация о том, как функционирует протокол, как запустить сервер Telnet во FreeBSD и как ограничить доступ к серверу, чтобы свести риск к минимуму

Схема функционирования протокола

Telnet — очень простой протокол. Во FreeBSD соответствующий клиент называется `telnet` Он устанавливает соединение с портом 23 указанного сервера например

```
$ telnet bunyan threeroomco.com
```

Клиент связывает терминал или окно, в котором была введена команда, с портом 23 компьютера `bunyan`. Сервер принимает входящий запрос и переадресует его программе `login`, которая обрабатывает запросы на регистрацию в текстовом режиме

как описывалось в главе 6 (вообще говоря Telnet функционирует под управлением суперсервера, т.е. демона `inetd`; именно он принимает запрос и вызывает сервер Telnet). Программа `login` высылает клиенту приглашение `login:` и, если процесс регистрации завершен успешно, соединяет клиентскую программу с интерпретатором команд пользователя, используя сервер Telnet в качестве посредника

В протоколе Telnet данные не модифицируются и не интерпретируются, что позволяет использовать клиент Telnet при изучении работы других серверов. Для этого достаточно указать номер порта в команде `telnet`. Например, команда `telnet bunyan.threeroomco.com 80` подключается к порту 80 удаленной системы. Таким образом, можно узнать, работает ли соответствующий сервер на удаленном компьютере и даже выполнить его отладку, если вы знаете низкоуровневые команды сервера.

Запуск сервера Telnet

Во FreeBSD сервер Telnet вызывается через демон `inetd`. Строка конфигурационного файла `/etc/inetd.conf`, описывающая запуск сервера, выглядит так

```
telnet  stream  tcp  nowait  root  /usr/libexec/telnetd  telnetd
```

По умолчанию она является комментарием, т.е. ей предшествует символ `#`. Чтобы активизировать сервер Telnet, выполните следующие действия:

- удалите символ комментария из строки, касающейся сервера Telnet;
- ееедите `killall -HUP inetd`, чтобы заставить демон `inetd` повторно прочи́тать конфигурационный файл.

С этого момента компьютер сможет принимать Telnet-запросы от любого компьютера, если только они не блокируются с помощью описанных ниже средств.

Ограничение доступа к серверу

Поскольку в Telnet данные не шифруются, протокол лучше всего применять в локальных сетях с небольшим числом пользователей. Применение протокола в Internet сопряжено с немалым риском из-за большого числа промежуточных систем. Существует целый ряд мер, с помощью которых можно ограничить доступ к серверу Telnet.

- **Сконфигурируйте механизм NAT на маршрутизаторе.** Система NAT (Network Address Translation — трансляция сетевых адресов), описанная в главе 17, позволяет компьютерам локальной сети быть представленными в Internet под общим IP-адресом. Вследствие этого внешние пользователи не могут обратиться к внутренним серверам напрямую, если только маршрутизатор NAT не разрешает непосредственный доступ к внутреннему серверу.
- **Задайте правила фильтрации на маршрутизаторе.** Маршрутизатор, сконфигурированный в качестве брандмауэра, защищает сеть от несанкционированных попыток внешнего доступа. В частности, можно ограничить доступ к порту 23 внутренних компьютеров, сделав сервер Telnet недоступным извне. Если конкретным внешним пользователям нужен Telnet-доступ в локальную сеть, задайте исключения из правил фильтрации.
- **Используйте локальные правила фильтрации.** Правила фильтрации могут быть реализованы и на локальном компьютере, чтобы разрешить доступ к Telnet-порту только авторизованным системам. Это эффективное средство ограничения внутреннего доступа к серверу.

- **Используйте программу TCP Wrappers.** Программа TCP Wrappers, подробно описанная в главе 29, реализует ограничения, напоминающие правила фильтрации. В стандартной конфигурации FreeBSD установки программы TCP Wrappers применяются ко всем серверам, запускаемым через демон `inetd`, хотя по умолчанию доступ не ограничивается.
- **Контролируйте использование учетных записей.** Убедитесь в том, что на компьютере, где работает сервер Telnet, нет неиспользуемых или ненужных учетных записей, и обеспечьте регулярную смену паролей. Неиспользуемые учетные записи часто становятся мишенью для атак, а регулярное изменение паролей позволяет сократить срок в течение которого у взломщика будет возможность подобрать пароль.

Средства ограничения доступа обычно основаны на проверке IP-адресов. Однако программа TCP Wrappers, например, может проверять имя вызывающего пользователя, если на клиентском компьютере работает сервер `ident`. На сервере реализуется еще одно ограничение, в стандартной конфигурации FreeBSD пользователю `root` запрещено регистрироваться в системе по протоколу Telnet. Это усложняет задачу потенциальному взломщику, узнавшему пароль суперпользователя, поскольку ему придется еще узнать пароль рядового пользователя, члена группы `wheel`. Только тогда он сможет зарегистрироваться в системе и выполнить команду `su`, чтобы стать суперпользователем.

Не рекомендуется выполнять команду `su` в сеансе Telnet, как и в любом другом незашифрованном соединении, поскольку пароль суперпользователя станет доступен любому анализатору пакетов, работающему в сети.

Конфигурирование сервера SSH

Протокол Telnet долгое время пользовался немалой популярностью. Он легко конфигурируется и в некоторых ситуациях очень удобен, хотя это не лучший протокол дистанционной регистрации. Передача данных в незашифрованном виде — слишком рискованное занятие в большинстве сетей. Существуют более безопасные варианты Telnet, например клиент и сервер, входящие в пакет Kerberos, но гораздо большую популярность в последнее время приобрел другой протокол — SSH. Его программное обеспечение сложнее инсталлировать и конфигурировать, однако сам протокол обладает целым рядом преимуществ по сравнению с Telnet.

Преимущества SSH

Простота Telnet контрастирует со сложностью SSH. В Telnet данные между клиентом и сервером передаются в незашифрованном виде без каких-либо изменений, тогда как в SSH все данные шифруются, а значит, сетевой трафик не поддается прямому анализу. Одно это делает SSH более предпочтительным протоколом дистанционной регистрации, чем Telnet, с точки зрения безопасности, поскольку пароли и конфиденциальные данные гораздо надежнее защищены от взлома. Как же функционирует протокол SSH?

Чтобы не пускаться в пространные объяснения, скажем, что в SSH применяются два вида шифрования, причем одно шифрование необходимо для инициализации второго. В первом случае используется *шифрование с открытым ключом*, когда одна из систем генерирует *открытый ключ* и *личный ключ* (ключ — это длинная цепочка цифр

предназначенная для кодирования и декодирования данных) Открытый ключ сообщается любому кто об этом попросит. Получатель открытого ключа шифрует с его помощью данные, высылаемые владельцу личного ключа. Только личный ключ позволяет дешифровать данные Имея открытый ключ можно также проверить подлинность данных, "подписанных" с помощью личного ключа.

Шифрование с открытым ключом применяется для установления начального соединения, но соответствующие алгоритмы требуют очень интенсивных вычислений со стороны центрального процессора, поэтому они используются только при согласовании *общего секретного ключа* для алгоритма симметричного шифрования Такой ключ обеспечивает как шифрование, так и дешифрование данных и должен быть известен только клиенту и серверу SSH. Он генерируется очень быстро и безопасно передается по сети благодаря алгоритму шифрования с открытым ключом

Все эти детали в основном скрыты от пользователей (многие клиенты SSH информируют пользователей о подключении к системе, с которой никогда раньше не взаимодействовали, это служит мерой предосторожности на тот случай, если злоумышленник попробует переадресовать входящий запрос на другой компьютер). Для пользователя работа по протоколу SSH мало чем отличается от применения Telnet (клиенты Telnet и SSH описаны в главе 24). Различие лишь одно: операции шифрования и дешифрования, обязательные в SSH, вызывают определенное снижение производительности, хотя обычно это не играет особой роли Более того, в соединениях текстового режима разница в производительности может быть вообще незаметна и проявляться лишь при туннелировании данных других протоколов

В Telnet имя пользователя и пароль запрашиваются программой `login` а в SSH они обрабатываются непосредственно клиентом и сервером, поэтому клиент SSH используется чуть иначе, чем клиент Telnet Это также означает, что для протокола SSH необходимо конфигурировать собственный модуль PAM Изменение настройки модуля PAM для программы `login` не отражается на работе SSH

В отличие от Telnet, в SSH поддерживается туннелирование других протоколов В этом случае клиент или сервер SSH начинает работать как сервер заданного протокола, но вместо того чтобы обрабатывать данные, он пересылает их на противоположный конец соединения, где соответствующая программа формирует запрос к локальному серверу. Чаще всего такая возможность используется для туннелирования X соединений, как описано на с 415, но могут обрабатываться и другие протоколы

Выбор сервера SSH

Во FreeBSD протокол SSH реализуют две программы

- **SSH.** Исходная программа SSH является коммерческим продуктом, доступным на Web-узле `www.ssh.com`. Однако для ознакомления и в определенных некоммерческих целях программу можно инсталлировать в виде пакета или переносимого приложения. Подробности описаны в лицензионном соглашении
- **OpenSSH.** Пакет OpenSSH (`www.openssh.com`) распространяется с открытым исходным кодом и представляет собой реализацию SSH выполненную для родственной операционной системы OpenBSD. В последних версиях FreeBSD этот пакет инсталлируется по умолчанию

OpenSSH — наиболее популярный сервер SSH для FreeBSD Он является частью стандартной инсталляции FreeBSD 5 0 и потому выбран для последующего описания, хотя применение исходного пакета SSH не сильно отличается Термин "SSH" будет использоваться для обозначения как самого протокола, так и обоих пакетов.

Помимо сервера нужно иметь в наличии клиент SSH. В пакете OpenSSH он называется `ssh`. Существуют клиенты и для других операционных систем, хотя они и не так распространены, как клиенты Telnet. Информацию о бесплатных клиентах SSH можно найти на Web-узле `www.freessh.org`. Коммерческие терминальные программы для Windows и Mac OS тоже могут поддерживать SSH.

Запуск сервера SSH

Один из способов запуска сервера SSH — воспользоваться программой `sysinstall`. Вот как это делается:

1. Перейдите в режим суперпользователя и введите **`/stand/sysinstall`**, чтобы запустить конфигурационную программу.
2. Выберите в основном меню программы пункт `Configure`.
3. Выберите в появившемся меню `FreeBSD Configuration Menu` пункт `Networking`.
4. В появившемся меню `Network Services Menu` установите флажок `Sshd`.
5. Последовательно выберите `OK`, `Cancel` и `Exit Install`, чтобы выйти из программы.

"За кулисами" в файл `/etc/rc.conf` добавляется следующая строка:

```
sshd_enable="YES"
```

Переменная `sshd_enable` определяет, должен ли на этапе начальной загрузки системы вызываться демон `sshd`. Эту строку можно добавить самостоятельно с помощью любого текстового редактора. Демон будет активизирован при очередном запуске системы. Если это нужно сделать немедленно, введите такую команду:

```
# /usr/sbin/sshd
```

В отличие от Telnet, сервер SSH обычно работает автономно, а не через суперсервер `inetd`. Такая конфигурация способствует повышению производительности, поскольку при каждом запуске демону `sshd` приходится создавать ключи шифрования. Параметры конфигурации демона хранятся в файле `sshd_config`, который описан в следующем разделе. Определенные установки файла разрешается переопределять с помощью аргументов командной строки демона. Это может понадобиться при отладке сервера.

Настройка параметров сервера

Конфигурационный файл сервера SSH называется `/etc/ssh/sshd_config`. Существует также клиентский конфигурационный файл `ssh_config`, находящийся в том же каталоге. Не путайте их! Файл `sshd_config` состоит из набора строк следующего формата:

```
Параметр значение
```

Название параметра является описательным, например `Port` или `PermitRootLogin`. В качестве значения может стоять число, константа `yes` или `no`, имя файла или произвольное строковое выражение, в зависимости от параметра. Строка, начинающаяся с символа '`#`', является комментарием. Как правило, стандартный файл `sshd_config` содержит вполне приемлемые установки. Перечислим параметры, которые чаще всего приходится настраивать:

- **Port** — задает порт прослушиваемый демоном sshd. Значение по умолчанию — 22, но можно задать другой порт, чтобы наличие сервера SSH стало менее очевидным.
- **ListenAddress** — заставляет демон sshd принимать запросы только через указанный интерфейс Интерфейс определяется по имени или IP-адресу, за которым после двоеточия может следовать номер порта например ListenAddress 172.20.30.40:2222. Это особенно удобно в системах с несколькими сетевыми интерфейсами
- **PermitRootLogin** — имеет значение yes или no и определяет, должен ли сервер SSH принимать запросы непосредственно от пользователя root. Установка по умолчанию no — является более безопасной
- **X11Forwarding** Сервер SSH может туннелировать данные любого протокола но для X Window System предусмотрены специальные средства поддержки, которые включаются с помощью параметра X11Forwarding, принимающего значение yes или no Установка по умолчанию yes, но значение no является более предпочтительным, если удаленные пользователи не должны иметь доступ к X-программам на сервере SSH
- **RhostsAuthentication** В SSH поддерживается старый метод аутентификации пользователей на основе файлов надежных узлов (/etc/hosts.equiv, $HOME/.rhosts и др), применяемый утилитой rlogin. В современных условиях использование этого метода весьма небезопасно, поэтому параметр RhostsAuthentication должен быть равен no
- **Subsystem**. С помощью этого параметра включается поддержка расширенных возможностей сервера. Он необычен тем, что принимает два значения' название протокола и путь к серверу этого протокола. Во FreeBSD сервер SSH содержит подсистему sftp, которая используется для передачи файлов через зашифрованный SSH туннель

Файл sshd config содержит множество других конфигурационных параметров Часть из них описана в самом файле, а остальные — на man-странице демона sshd

Обратите внимание на параметр ListenAddress В системах с несколькими сетевыми интерфейсами рекомендуется, чтобы сервер прослушивал только один из них. Например, маршрутизатор можно сконфигурировать так чтобы он принимал SSH запросы из локальной сети, но не из Internet.

Если в файл sshd config внесены изменения, нужно перезапустить демон sshd чтобы они вступили в силу. Вот как это делается·

```
# killall sshd
# /usr/sbin/sshd
```


Первая из этих команд разрывает все существующие SSH-соединения. Чтобы избежать этого, выполните команду ps ax | grep sshd и выясните, какой из процессов sshd является родительским. Его легко узнать по отсутствию имени пользователя или номера терминала

Настройка удаленного доступа в X Window System

Посредством Telnet или SSH можно запускать многие программы FreeBSD, в ча стности интерпретаторы команд, Emacs, mutt и даже утилиты системного администрирования Однако программы конечных пользователей, как правило, имеют графи ческий интерфейс и работают в X-среде Впрочем, дистанционный запуск таких про грамм тоже возможен, поскольку X среда проектировалась и как среда удаленного доступа. Модель взаимодействия клиента и сервера X Window System достаточно непривычна для новичка и требует предварительного изучения. В ней поддерживаются два вида удаленного доступа посредством сервера дистанционной регистрации, работающего в текстовом режиме, и посредством специальной графической программы регистрации Оба этих метода описаны ниже

Взаимодействие клиента и сераера X Window System

Принято считать, что серверы — это высокопроизводительные системы размещаемые в отдельных комнатах с ограниченным доступом и контролируемые администраторами В таком случае клиенты это менее производительные системы за которыми работают пользователи Конечно, зачастую все именно так и есть. Пользователи непо средственно работают с почтовыми, FTP- и Web-клиентами, а соответствующие серверы являются действительно мощными системами, находящимися в отдельных помещениях Но когда речь заходит об X-серверах, составляющих ядро X Window System, традиционная модель взаимодействия "клиент - сервер" оказывается неуместной. Чтобы понять, почему это так, рассмотрим ситуацию с точки зрения клиент ской программы FTP клиент, например, устанавливает соединение с FTP-сервером, чтобы запросить передачу данных Если FTP-клиент вызывается с системной консо ли, т е в текстовом режиме, то компьютер, за которым работает пользователь, является клиентом, как и предполагает стандартная модель Но что если FTP-клиент работает в графическом режиме? Для такой программы экран, клавиатура и мышь представляются сетевыми устройствами ввода-вывода, как и удаленный FTP сервер. FTP-клиент является одновременно и клиентом X-сервера, который передает ему входные данные от клавиатуры и мыши и позволяет посылать данные для отображения на экране. Тот факт, что за клавиатурой сидит пользователь, FTP-клиенту безразличен. Схематически эта модель взаимодействия изображена на рис 21 1. Часто X клиент и X-сервер работают на одном компьютере, но это вовсе не обязательно

Рис 21 1 Графический FTP-клиент взаимодействует как с FTP сервером, так и с X-сервером, который может функционировать на том же самом или другом компьютере

В стандартной инсталляции FreeBSD система сконфигурирована таким образом что программы взаимодействуют с X-сервером, работающим на том же компьютере В случае удаленного доступа нужно изменить конфигурацию программ, заставить X-сервер принимать запросы от клиентской системы и найти способ устанавливать соединение с клиентской системой

В этой главе в качестве X-сервера используется FreeBSD (приводимая информация справедлива для любой системы, в которой работает программа XFree86). Но программы FreeBSD могут вызываться и из другой операционной системы, например Windows или Mac OS. Для этого необходимо соответствующее программное обеспечение В качестве примеров назовем XFree86 (перенесена в Windows - `http://www.cygwin.com/xfree/`; в OS/2 - `ftp://ftp.xfree86.org/pub/XFree86/3.3.6/binaries/OS2/`; в Mac OS X - `http://mrcla.com/XonX/`), Xmanager for Windows (`http://www.netsarang.com/products/xmanager.html`), Exceed for Windows (`http://www.hummingbird.com/products/nc/exceed/`) и Xtools for Mac OS X (`http://www.tenon.com/products/xtools/`). Большая часть материала этой главы касающаяся удаленного доступа в X-среде, применима и к этим продуктам, хотя некоторые детали конфигурации X-сервера будут, естественно, отличаться

X-серверы работают в одном из двух режимов: *управляющем* или *смешанном* Управляющий сервер формирует рабочий стол Это может быть полноэкраннал среда или просто окно, в котором X-сервер создает подчиненные окна клиентских программ В смешанной среде окна удаленного X-клиента отображаются наряду с окнами локальной системы. Как правило, во FreeBSD сервер XFree86 работает в смешанном режиме, поэтому локальные окна сложно отличить от окон удаленной системы, зато их проще упорядочить. Многие X-серверы Windows и Mac OS поддерживают оба режима, так что при желании можно изолировать X-среду от графической среды базовой операционной системы. Эта изоляция важна, если нужно запустить графическую оболочку типа GNOME или KDE.

Регистрация в X-среде в текстоаом режиме

Поскольку пользователь работает за X-сервером, ему необходимо каким то образом обратиться к клиентскому компьютеру с просьбой о запуске нужной X-программы. Это означает, что на клиентском компьютере должен функционировать сервер, разрешающий удаленный доступ Для этой цели вполне подойдут серверы удаленного доступа функционирующие в текстовом режиме в частности Telnet и SSH. Некоторые из них, например SSH имеют средства упрощающие доступ в X-среде

Базовая процедура

В простейшем случае достаточно иметь локальный X-сервер, локальную утилиту регистрации в текстовом режиме и соответствующий сервер дистанционной регистрации, а также знать несколько основных команд Ниже описана последовательность действий, которую нужно выполнять для запуска программ удаленной UNIX-системы (`bunyan.threeroomco.com`) из другой UNIX-системы (`blueox.threeroomco.com`).

1. Зарегистрируйтесь на компьютере `blueox.threeroomco.com` и если необходимо, загрузите X-среду (например, можно ввести **`startx`**)
2. В окне `xterm` введите **`xhost +bunyan.threeroomco.com`**. Эта команда позволит X-серверу `blueox` принимать запросы от компьютера `bunyan` Доменное имя можно опустить, если оба компьютера находятся в одном домене. Вместо имен компьютеров разрешается указывать IP-адреса

3 Воспользуйтесь выбранным средством удаленного доступа таким как Telnet или SSH, для регистрации на компьютере bunyan

4 На компьютере bunyan выполните команду, которая сделает переменную среды DISPLAY равной blueox.threeroomco com.0 В интерпретаторе sh или bash эта команда выглядит так: **export DISPLAY=blueox.threeroomco.com:0** В интерпретаторе tcsh команда такова: **setenv DISPLAY blueox.threeroomco.com:0**. Опять-таки, можно опустить доменное имя или указать IP адрес.

5 На компьютере bunyan введите имя нужной программы. Устройством отображения ее данных станет монитор компьютера blueox Если, к примеру, запустить программу xterm то на экране компьютера blueox появится второе терминальное окно, только теперь оно будет контролироваться компьютером bunyan. Подобным образом можно запускать сколько угодно программ

Рассмотренная процедура применяется во многих ситуациях. Обычно она имеет место в диспетчере окон или в графической оболочке, которые контролируются X-сервером (blueox) Если же при загрузке X-среды диспетчер окон не запускается, его можно вызвать на этапе 5, только он будет контролироваться удаленной системой.

Безопасность имени пользователя и пароля определяется протоколом дистанционной регистрации, применяемым на этапе 3. Но сами данные, передаваемые X-программой, никак не защищены. Например, если запустить на компьютере bunyan программу xterm и выполнить в ней команду su, то вводимый пароль будет передан по сети в незашифрованном виде, даже если на этапе 3 использовался протокол SSH. В то же время в окне xterm из которого на этапе 3 устанавливалось соединение, та же самая команда su будет выполняться вполне безопасно.

Если между взаимодействующими компьютерами находится промежуточный маршрутизатор или брандмауэр для него может понадобиться специальная конфигурация. К примеру, когда компьютер обращается к удаленной системе, находящейся в другой сети, маршрутизатор NAT пропускает начальный запрос на регистрацию (этап 3) Однако обратный запрос, выполняемый при запуске X-программы (этап 5), тоже поступает на маршрутизатор, который, следовательно, должен быть сконфигурирован таким образом, чтобы перенаправлять трафик порта 6000 X-серверу

Приведенные инструкции предполагают наличие одного X-сеанса Если на локальном компьютере несколько таких сеансов, то на этапе 4 нужно после двоеточия указать номер соответствующего сеанса Первому сеансу соответствует номер 0, второму — 1 и т.д

В случае, когда X-сервер функционирует в Windows или в другой операционной системе, не относящейся к семейству UNIX, этап 2 можно пропустить: большинство таких серверов по умолчанию принимает запросы от любых систем При этом возникает проблема: команда xhost разрешает серверу обрабатывать запросы от любого пользователя удаленной системы. Таким образом, если в системе много пользователей, кто-нибудь из них может открывать ненужные окна и даже перехватывать нажатия клавиш, чтобы узнать чужой пароль. Решения этой проблемы описаны ниже

Туннелирование X-трафика средствами SSH

Одна из модификаций рассмотренной выше процедуры предполагает использование протокола SSH для туннелирования X-трафика В такой схеме SSH-сервер удаленной системы (bunyan) конфигурирует себя так будто он является локальным X-сервером Этот "псевдосервер" функционирует под видом X-сеанса номер 10 (bunyan.threeroomco.com:10) и автоматически устанавливает корректное значение переменной среды DISPLAY.

Когда пользователь запускает X-программу SSH-сервер принимает X трафик и перенаправляет его SSH-клиенту, который, в свою очередь, связывается с локальным X-сервером и передает трафик ему. Ответные данные от локального X-сервера посту пают SSH-клиенту, который пересылает их SSH-серверу, а тот — X-клиентам (запущенным программам)

С практической точки зрения такая конфигурация означает, что в приведенной выше процедуре регистрации нужно пропустить этапы 2 и 4. Поскольку X-трафик туннелируется, этот метод будет работать и при наличии брандмауэра или маршрутизатора NAT, только необходимо, чтобы это устройство разрешало исходящие SSH соединения

Режим туннелирования X-трафика должен быть сконфигурирован как на сервере, так и на клиенте SSH. Ранее уже упоминалась соответствующая серверная опция: `X11Forwarding` Аналогичная установка есть и в клиентском файле `/etc/ssh/ssh_config`. Она должна выглядеть так

```
ForwardX11 yes
```

То же самое можно сделать и с помощью опций командной строки `-X` и `-x` Если сервер не сконфигурирован на переадресацию X-соединений придется выполнить всю последовательность действий, описанную на с 414

Преимуществом использования протокола SSH является не только устранение сложностей, связанных с конфигурированием брандмауэров и маршрутизаторов NAT, но и, самое главное, шифрование всего X-трафика. Можно вводить пароли и отобра жать конфиденциальную информацию, не беспокоясь о том, что данные попадут в чужие руки (теоретически протокол SSH можно "взломать", однако риск очень невелик) Впрочем, за удовольствие приходится платить шифрование отнимает немало времени. Производительность X-приложений при работе поверх SSH будет ниже, чем при непосредственном вызове. На помощь может прийти опция `-C` утилиты `ssh`, кото рая дополнительно включает режим сжатия данных. Это позволяет передавать больше данных при наличии сетевого канала с ограниченной пропускной способностью

Применение утилиты xauth

Если протокол SSH по каким либо причинам не используется, возникают упоми навшиеся ранее проблемы с безопасностью В частности, любой пользователь удаленного компьютера получает доступ к вашему X-дисплею, что чревато серьезными не приятностями Для уменьшения риска необходимо заменить команду `xhost` другой утилитой, позволяющей указывать имена разрешенных пользователей. Одна из таких утилит — `xauth` Она использует ключи, напоминающие ключи SSH, но они всего лишь аутентифицируют клиентскую программу как такую которой разрешено рабо тать с ресурсами X-сервера Процедура регистрации будет выглядеть так же, как и в предыдущем случае, только на этапе 2 необходимо выполнить следующую команду

```
$ xauth list blueox:0 | sed -e 's/^/add /' | ssh bunyan -x xauth
```

Эта команда извлекает ключ программы `xauth`, соответствующий X-серверу `blueox`, и преобразует его в аргумент командной строки, который посылается утилите `xauth` X-клиента, чтобы ключ стал доступен клиентским программам. При этом появится запрос на ввод пароля в удаленной системе. Последнее действие выполняется утилитой `ssh` Это значит, что на компьютере X-клиента должен работать сервер SSH, а на компьютере сервера необходимо присутствие клиента SSH. В такой ситуации лучше туннелировать X-трафик средствами SSH, т.е. вообще обойтись без утилиты `xauth` Если же пакет SSH не инсталлирован ни в одной из систем, можно найти другие способы передачи ключа Например, если на компьютере X-клиента работает демон `rshd`, введите **`rsh bunyan xauth`** вместо **`ssh bunyan -x xauth`** К сожалению демон `rshd` далеко не так безопасен, как `sshd` поэтому не может считаться

удачной альтернативой. Более сложный вариант заключается в записи ключа в файл с последующим копированием файла в удаленную систему и добавлением ключа в файл X идентификаторов

Утилита `xauth` не слишком привлекательна для использования из-за трудностей связанных с передачей ключа, к тому же не выполняется шифрование данных. Она впрочем, удобна в некоторых ситуациях, например когда туннелирование трафика средствами SSH приводит к существенному снижению производительности. Утилита `xauth` используется также другими программами, в частности теми, что описаны в следующем подразделе. Но этот факт скрыт и от пользователя и от администратора поэтому никакого дополнительного конфигурирования не требуется.

Регистрация по протоколу XDMCP

Иногда доступ к компьютеру X-клиента в текстовом режиме неудобен или невозможен. К примеру, *X-терминалы* — это выделенные графические терминалы, позволяющие множеству пользователей совместно работать за одним мощным компьютером. Как правило, они не оснащены клиентами Telnet или SSH. В качестве X-терминала можно сконфигурировать даже старую систему семейства *x86*. Кроме того, пятиэтапная процедура регистрации, приведенная на с. 414, слишком неудобна для рядовых пользователей, предпочитающих работать в графическом режиме. Решением проблемы является XDMCP (X Display Manager Control Protocol) - протокол дистанционной регистрации, предназначенный для X-среды. Сервер XDMCP заменяет собой сервер Telnet или SSH в процедуре регистрации, автоматизируя многие ее этапы. Чтобы этот сервер применялся, необходимо соответствующим образом сконфигурировать X-клиент. Функции клиентов XDMCP встроены в большинство X-серверов. К сожалению, в протоколе XDMCP имена пользователей и пароли не шифруются, так что он не более безопасен, чем Telnet.

На заметку

Если FreeBSD-система сконфигурирована на загрузку в графическом режиме, как описывалось в главе 6, то протокол XDMCP уже используется. Нужно лишь внести некоторые изменения в конфигурацию системы, чтобы она принимала запросы от дистанционных X-серверов.

Конфигурирование сервера XDMCP

Во FreeBSD поддерживаются четыре сервера XDMCP:

- **XDM** (X Display Manager). Самый первый член семейства экранных диспетчеров является наиболее простым из серверов XDMCP. Он отображает приглашение на ввод имени пользователя и пароля и не разрешает пользователю устанавливать дополнительные параметры на этапе регистрации. Все необходимые установки хранятся в сценарии `.xsession`, который находится в начальном каталоге пользователя (см. главу 6).
- **WDM** (WING Display Manager) — расширение сервера XDM. Программа WDM позволяет устанавливать различные параметры на этапе регистрации, такие например, как запускаемая графическая оболочка. Формат конфигурационных файлов WDM и XDM во многом сходен.
- **KDM** (KDE Display Manager) — является частью графической оболочки KDE. Конфигурационный файл программы имеет тот же формат, что и в XDM, но дополнительные опции, такие как список графических оболочек, выбираемых по запросу пользователя, задаются во вспомогательном конфигурационном файле.

- GDM (GNOME Desktop Manager) — часть графической оболочки GNOME Подобно WDM и KDM, программа позволяет задавать расширенные параметры на этапе регистрации, но использует совершенно иной формат конфигурационного файла

Поскольку у первых трех программ форматы конфигурационных файлов примерно одинаковы, именно их настройка (на примере XDM) будет описана ниже. Программа GDM подключается сходным образом, а в документации приведена информация о том, как заставить ее принимать запросы от удаленных систем. К программе прилагается также графическая утилита `gdmconfig`, помогающая задавать многие ее параметры.

Первое, что нужно сделать для разрешения дистанционных запросов, — активизировать сервер. Это осуществляется путем редактирования файла `/etc/ttys`, содержащего следующую строку:

```
ttyv8   "/usr/X11R6/bin/xdm -nodaemon"  xterm   off secure
```

Поменяйте `off` на `on`, и сервер будет подключен. Вместо `/usr/X11R6/bin/xdm` можно указать путь к другому серверу XDMCP, например `/usr/X11R6/bin/wdm`. Чтобы сервер начал работу, нужно либо перезагрузить компьютер, либо ввести **`kill -HUP 1`**: эта команда заставит демон `init` перечитать свои конфигурационные файлы. В результате появится экран регистрации программы XDM. Однако дистанционная регистрация по протоколу XDMCP еще не будет разрешена. Для этого необходимо сделать дополнительные изменения.

Конфигурационный файл программы XDM называется `xdm-config`. Во FreeBSD 5.0 он хранится в каталоге `/usr/X11R6/lib/xdm`, но в более ранних версиях FreeBSD его местонахождение может быть иным, например `/etc/X11/xdm`. Стандартный файл `xdm-config` содержит следующую строку (расположена ближе к концу файла):

```
DisplayManager.requestPort      0
```

Эта строка запрещает программе XDM опрашивать порт, через который поступают дистанционные запросы на регистрацию. В результате серверу остается управлять лишь локальным дисплеем. Чтобы разрешить обработку запросов, нужно либо превратить эту строку в комментарий, поставив в ее начало знак '#', либо указать в ней реальный номер порта, например 177 (стандартный порт XDMCP).

Еще одно изменение является обязательным: нужно отредактировать файл `Xaccess`, находящийся в том же каталоге, что и `xdm-config`. Этот файл определяет, какие компьютеры могут обращаться к серверу XDM и на запросы каких типов он должен отвечать. Чтобы разрешить весь спектр запросов, добавьте следующие строки:

```
CHOOSER BROADCAST
```

Первая строка говорит о том, что любой компьютер может запросить доступ в систему. Вторая строка заставляет программу осуществлять широковещательную рассылку запроса другим серверам XDMCP и предоставлять список найденных компьютеров запрашивающей системе (это называется *непрямым запросом*). Если в сети есть компьютеры, на которых программа XFree86 работает в режиме X-терминала, то по крайней мере одна из систем должна содержать вторую из показанных строк, чтобы X-терминалы могли определять, какие компьютеры поддерживают протокол XDMCP. Некоторые X-серверы, особенно написанные для Windows, самостоятельно находят серверы XDMCP, поэтому им не нужно посылать непрямые запросы.

Звездочки в обеих строках примера означают "любой компьютер". Чтобы ограничить список компьютеров, укажите их по одному в строке (звездочка может заменять часть имени). Рассмотрим следующие записи:

```
*.threeroomco.com
pine.pangaea.edu
```

```
pine.pangaea.edu CHOOSER BROADCAST
bunyan.threeroomco.com CHOOSER BROADCAST
```

В первом случае компьютерам домена `threeroomco.com` разрешено получать приглашение на регистрацию по протоколу XDMCP. Компьютеру `pine.pangaea.edu` предоставлено то же самое право, кроме того, он может получать список других серверов XDMCP посредством непрямого запроса. Непрямые запросы разрешено выполнять и компьютеру `bunyan.threeroomco.com`. Такого рода запросы особенно важны для клиентов, которым недоступен широковещательный режим. Широковещательные запросы могут не проходить через Internet, и даже в локальных сетях бывают ограничения на широковещательную рассылку.

Обычно сервер XDMCP управляет локальным дисплеем, т. е. он выступает в качестве клиента для соответствующего X-сервера. Такая конфигурация задана в файле `Xservers`, который находится в том же каталоге, что и `xdm-config` и `Xaccess`. По умолчанию он содержит следующую строку:

```
:0 local /usr/X11R6/bin/X
```

Она заставляет сервер управлять первым (`:0`) дисплеем локального компьютера, вызывая для этого программу `/usr/X11R6/bin/X`. Если же необходимо, чтобы X-приложения могли вызываться с дистанционного сервера, и при этом на локальном компьютере X-среда не должна запускаться, превратите показанную строку в комментарии с помощью символа '#'. Тогда сервер сможет отвечать на дистанционные запросы, не запуская автоматически X-среду.

После внесения изменений в файлы `xdm-config`, `Xaccess` и `Xservers` выполните команду **`kill -HUP 1`**, чтобы заставить демон `init` переконфигурировать выполняющиеся процессы. В результате будет прочитан файл `/etc/ttys` и загрузится программа XDM, если до этого она отсутствовала. В противном случае, не исключено, придется предварительно уничтожить программу, введя **`killall xdm`**. Демон `init` перезапустит программу с новыми параметрами, заданными в файле `/etc/ttys`, а программа загрузит измененные конфигурационные файлы.

Использование клиента XDMCP

Клиенты XDMCP обычно встроены в X-серверы или тесно связаны с ними. Особенности их конфигурирования и применения зависят от конкретного сервера и описаны в документации к нему. В Windows X-сервер отображает диалоговое окно настроек, в котором можно задать использование протокола XDMCP для дистанционной регистрации. На рис. 21.2 приведено окно программы Xmanager (`www.netsarang.com`). В нем представлены четыре типичных параметра XDMCP-клиента.

- **Пассивный режим**. В пассивном режиме X-сервер не пытается устанавливать XDMCP-соединения. Этот режим используется, если предусмотрена регистрация в текстовом режиме или XDMCP-сервер должен самостоятельно связываться с X-сервером. Такой режим применяется, когда во FreeBSD включен режим графической регистрации в системе.
- **Запрос XDMCP**. В этом случае X-сервер будет регистрироваться непосредственно на компьютере сервера XDMCP. Пользователи будут лишены возможности выбрать другой компьютер.
- **Широковещательный запрос XDMCP**. В этом режиме X-сервер посылает широковещательный запрос в локальную сеть или заданной группе компьютеров, чтобы выяснить, на каких компьютерах доступны серверы XDMCP. Собрав необходимую информацию, X-сервер отображает список доступных серверов (рис. 21.3). Из этого списка пользователь может выбрать нужную систему, чтобы зарегистрироваться в ней.

- **Непрямой запрос XDMCP** Для пользователя непрямой запрос выглядит как широковещательный Но в данном случае необходимо сконфигурировать X-сервер так, чтобы он использовал один из XDMCP-серверов в качестве посредника Соответствующий XDMCP-сервер, в свою очередь, должен быть сконфигурирован на прием непрямых запросов, как описывалось выше Полученный список систем будет выглядеть примерно так, как на рис 21 3

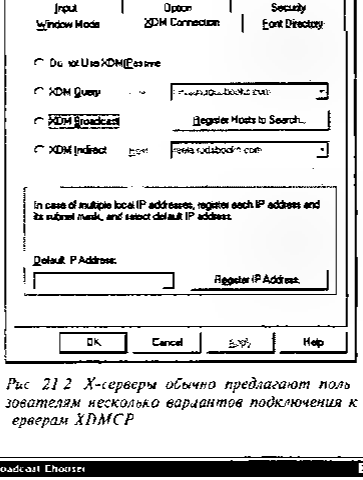


Рис 21 2 X-серверы обычно предлагают пользователям несколько вариантов подключения к серверам XDMCP

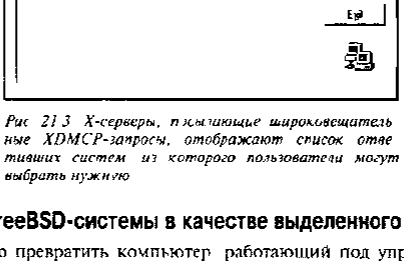


Рис 21 3 X-серверы, посылающие широковещательные XDMCP-запросы, отображают список ответивших систем, из которого пользователи могут выбрать нужную

Использование FreeBSD-системы в качестве выделенного X-терминала

Если необходимо превратить компьютер, работающий под управлением FreeBSD в выделенный X-терминал, включите в системе правильный режим обработки XDMCP запросов Как ни странно система не должна автоматически загружать X-среду

Прежде всего нужно отключить собственный XDMCP-сервер системы Измените строку вызова программы `xdm` в файле `/etc/ttys` таким образом, чтобы сервер стал недоступен. Далее проверьте собственные функции XDMCP клиента программы XFree86 Они активизируются с помощью опций `-query` *имя_компьютера* (разрешить регистрацию на заданном компьютере), `-broadcast` (осуществлять широковещательную рассылку запросов на регистрацию) и `-indirect` *имя_компьютера* (направлять непрямой запрос на указанный компьютер) Вот, к примеру, как получить список доступных систем посредством непрямого запроса:

```
$ /usr/X11R6/bin/X -indirect blueox.threeroomco.com
```

Разумеется, на компьютере `blueox.threeroomco.com` должен функционировать сервер XDMCP, сконфигурированный на прием непрямых запросов от искомого компьютера Если это так и X-среда вызывающей системы сконфигурирована правильно, появится окно со списком найденных серверов, как на рис 21 3

На заметку

Программа XFree86 имеет одну особенность, отличающую ее от других X-серверов: опция `-broadcast` заставляет программу регистрироваться на первом из компьютеров, найденном в локальной сети, вместо того чтобы отображать окно выбора сервера Таким образом, нужно либо использовать опцию `-indirect`, либо сразу задать нужный компьютер с помощью опции `-query`.

Убедившись в том, что все работает как надо, автоматизируйте запуск X-среды Для этого нужно создать сценарий запуска, активизируемый на конечных стадиях загрузки системы, как описывалось в главе 6 Если воспользоваться опцией `-indirect` компьютер начнет функционировать как X-терминал: после загрузки система выдаст список доступных XDMCP-серверов, позволяя пользователю зарегистрироваться на любом из них (при условии что пользователь имеет соответствующие учетные записи)

Запуск сервера VNC

Среда X Window System проектировалась с расчетом на сетевой доступ, поэтому в ней можно работать из другой FreeBSD-системы, из другой UNIX-системы и даже из системы, не относящейся к семейству UNIX, при условии, что в этой системе инсталлирован X-сервер Однако и здесь не обходится без трудностей. Как рассказывалось в разделе "Настройка удаленного доступа в X Window System", проблемы чаще всего возникают, когда между X-сервером и компьютером, с которого запускаются программы, находится брандмауэр или маршрутизатор NAT Кроме того, большинство X-серверов для Windows и Mac OS являются коммерческими программами и стоят немало. Пакет XFree86 бесплатен, хотя его сложнее конфигурировать Наконец, программа XFree86 позволяет дистанционно выполнять X-приложения, но она не слишком подходит для ситуации, когда с FreeBSD-компьютера требуется запускать программы в среде Windows. Вот почему многие пользователи предпочитают применять другое графическое средство удаленного доступа: *VNC (Virtual Network Computing* — виртуальная среда сетевых вычислений) Это программа с открытым исходным кодом (`www.realvnc.com`), разработанная компанией AT&T Ее клиент, инсталлируемый в системе, где работает пользователь, контролирует специальный X-сервер на удаленном компьютере. В результате инсталляция, конфигурирование и применение VNC осуществляется совсем не так как в случае дистанционного X-сервера

Взаимодействие клиента и сервера VNC

Вернемся к рис. 21.1, на котором изображена схема взаимодействия X клиента (т е X-приложения) и X-сервера в сетевой среде. VNC добавляет еще один уровень сложности к этой схеме, организуя запуск X-сервера на том же компьютере, где работает X-клиент Этот сервер играет также роль VNC-сервера. На компьютере, который на рис 21 1 обозначен как X сервер, функционирует VNC-клиент, взаимодействующий с локальным X-сервером (в среде FreeBSD). В сущности, в этой схеме присутствуют два X-сераера, один из которых посылает копии своих данных по сети другому серверу, управляющему клавиатурой, мышью и экраном.

На заметку

Клиенты и серверы VNC доступны для Windows, Mac OS и других операционных систем. Если клиент или сервер функционирует не в среде FreeBSD (точнее, в среде, не совместимой с X Window System), то X Window System не задействуется. Вместо этого VNC использует системные вызовы для отображения данных, полученных от другой системы, или же перехватывает данные направляемые операционной системой на экран

Помимо добавления нового протокольного уровня, VNC повышает также сложность соединения. По иронии, именно эта сложность упрощает конфигурирование удаленного доступа. Суть в том, что для удаленного доступа в X-среде требуется, чтобы на обеих сторонах соединения присутствовал как клиент, так и сервер, а VNC берет на себя обе эти роли. VNC-клиент является клиентом дистанционной регистрации а заодно заменяет (или использует) локальный X-сервер. На противоположном конце соединения VNC-сервер функционирует как локальный X-сервер, что избавляет X-клиент от необходимости устанавливать обратное соединение с компьютером за которым работает пользователь Сходным образом действует и механизм переадресации X-запросов в SSH, хотя отображаемые данные там не обрабатываются. Стало быть, для SSH необходимо наличие X-сервера на компьютере SSH-клиента, тогда как VNC может обойтись без X-сервера на стороне клиента. Для VNC нужен лишь некий механизм управления экраном, такой как подсистема управления окнами в Windows или Mac OS

Что касается безопасности, то VNC шифрует регистрационный пароль, но не остальные данные. В этом смысле VNC безопаснее, чем Telnet или XDMCP, однако оптимальным решением все же является туннелирование X-трафика средствами SSH

Инсталляция VNC

Программа VNC доступна как в виде переносимого приложения, так и в виде пакета (в обоих случаях присутствует в разделе `net`). В процессе инсталляции создается несколько программных файлов, включая `vncpasswd` (устанавливает пароль, используемый сервером) `Xvnc` (собственно X-сервер и VNC-сервер), `vncserver` (Perl-сценарий, запускающий программу `Xvnc` со стандартными параметрами) и `vncviewer` (VNC-клиент) Для того чтобы получить доступ из FreeBSD к другому FreeBSD-компьютеру, необходимо инсталлировать весь пакет VNC на обоих компьютерах

Помимо исходного пакета VNC существуют и другие его разновидности. К примеру, TightVNC (`www.tightvnc.com`) и TridiaVNC (`www.developvnc.org`) обеспечивают сжатие данных, что ускоряет VNC-операции Некоторые из альтернатив доступны в виде пакетов или переносимых приложений FreeBSD, и большинство из них обратно совместимо с исходной версией VNC, поэтому им чаще отдают предпочтение Ниже будет описано конфигурирование базового варианта VNC

Серверы и клиенты VNC доступны для различных платформ даже таких, как OS/2 (http://www.sra.co.jp/people/akira/os2/vnc-pm/) и BeOS (http://www.bebits.com/app/2329) Информация о проектах для Windows и Mac OS доступна на Web-узле VNC

Конфигурирование сервера VNC

В отличие от большинства серверов дистанционной регистрации, каждый экземпляр сервера VNC связан с конкретным пользователем (на с 426 будет показано, как организовать запуск VNC-сервера более традиционным способом) Это значит, что конфигурированием сервера занимаются сами пользователи Конфигурационные файлы находятся в пользовательском каталоге ~/.vnc. Системный администратор может настроить сценарий запуска vncserver, задавая стандартные параметры сервера в соответствии с особенностями системы

Пользовательские модификации

Каталог ~/.vnc и файлы в нем создаются при запуске VNC сервера или вспомогательных утилит Для начала выполните следующие действия

1. Создайте подкаталог .vnc в начальном каталоге пользователя, который будет запускать VNC-сервер Убедитесь в том что именно этот пользователь является владельцем каталога
2. *От имени пользователя, запускающего сервер,* введите **vncpasswd** Подобно стандартной команде passwd, утилита vncpasswd задает пароль, сохраняемый в зашифрованном виде в файле ~/.vnc/passwd Утилита должна создать этот файл с правами доступа, разрешающими чтение файла только владельцу, но не полениться проверить полученный результат

С этого момента VNC-сервер готов к работе Запускать сервер нужно *от имени прежнего пользователя* (того, что вводил **vncpasswd**).

```
$ vncserver

New  X  desktop is bunyan.threeroomco.com:1

Creating default startup script /home/rodsmith/.vnc/xstartup
Starting applications specified in /home/rodsmith/.vnc/xstartup
Log file is /home/rodsmith/.vnc/bunyan.threeroomco.com:1.log
```

Теперь VNC-сервер активен, и на нем можно зарегистрироваться, как будет описано на с 425 Обратите внимание на номер VNC-сеанса, сообщаемый утилитой vncserver Он указан в первой строке выходных данных после двоеточия (1 в данном примере) Номер сеанса заменяет имя пользователя Если сервер вызывается двумя и более пользователями или же один пользователь запускает сервер несколько раз номер сеанса в каждом случае будет другим.

На заметку

Читатели, должно быть, удивлены как можно ввести **vncserver** для запуска VNC-сервера, сидя за другим компьютером? Одно из решений воспользоваться другим средством дистанционной регистрации, например SSH Если у вас есть физический доступ к удаленной системе, запустите VNC-сервер заранее — это упростит удаленный доступ На с 426 будет продемонстрировано другое решение

Поскольку VNC-сервер включает в себя X-сервер, для него требуется сценарий запуска X-среды Когда утилита vncserver вызывается впервые, она создает базовый вариант такого сценария ~/.vnc/xstartup. Этот сценарий активизирует X-сеанс с

использованием диспетчера окон twm. Можете отредактировать сценарий, чтобы запускалась другая графическая оболочка, например GNOME или KDE. В главе 6 описывалось, как вносить такие изменения в файлы .xinitrc и xsession. VNC сценарий xstartup работает точно так же.

Одной из необычных особенностей VNC является сохранение состояния сервера между сеансами. Предположим, например, что пользователь запускает VNC-сервер и работает с ним какое-то время. Затем он выгружает свой VNC-клиент, садится за другой компьютер и начинает новый сеанс связи с тем же VNC-сервером. В результате пользователь обнаружит рабочий стол в том состоянии, в каком он находился на момент завершения первого сеанса. Эту особенность удобно использовать в определенных ситуациях, но она же означает, что изменения в сценарии xstartup не вступят в силу до перезапуска сервера. Вот как это сделать:

```
$ vncserver -kill :1
```

Число после двоеточия соответствует номеру сеанса, о котором сообщается при запуске сервера. Задайте нужный номер и, после того как сервер перезагрузится, зарегистрируйтесь снова. Сервер следует отключать, если его использование в ближайшее время не планируется. Это особенно важно для компьютера, доступного через Internet, так как уменьшается риск взлома системы. Кроме того, в случае сбоя сервера (или компьютера, на котором он работает) вся несохраненная работа будет потеряна.

Модификация сценария vncserver

Сценарий /usr/X11R6/bin/vncserver устанавливает множество глобальных конфигурационных параметров VNC-сервера — примерно, как файл XF86Config задает глобальные параметры X-сервера, управляющего дисплеем. В частности, сценарий vncserver определяет размер виртуального рабочего стола VNC, цветовую разрядность и стандартный пользовательский сценарий запуска. Все эти параметры можно модифицировать путем непосредственного редактирования сценария vncserver. Рассмотрим их подробнее:

- **Размер рабочего стола**. Переменная $geometry задает размер виртуального рабочего стола, создаваемого VNC-сервером для пользователя. Значение по умолчанию — 1024x768.

Виртуальный рабочий стол VNC формируется в окне VNC клиента. Оптимальным является размер рабочего стола, чуть меньший клиентского экрана. Например, при разрешении экрана 1024x768 параметр $geometry можно задать равным 950x650.

- **Цветовая разрядность**. Стандартная разрядность цветовой палитры равна 8 бит (256 цветов). Если клиентская система поддерживает большую разрядность, то задайте значение 16, 24 или 32.
- **Путь к файлам шрифтов**. Стандартный путь к файлам шрифтов хранится в переменной $cmd. Чтобы задать новый путь, нужно вызвать сценарий с опцией -fp. По умолчанию проверяются каталоги /usr/lib/X11/fonts/misc и /usr/lib/X11/fonts/75dpi. Имена каталогов разделяются запятыми. Информация о каталогах шрифтов локального X сервера хранится в файле /etc/X11/XF86Config.
- **Стандартный сценарий запуска для новых пользователей**. Сценарии vncserver содержит код, записываемый в файл ~/vnc/xstartup при первом запуске сервера. Таким образом, чтобы модифицировать сценарий запуска, нужно отредактировать сценарий vncserver. Поищите переменную $defaultXStartup и сравните ее с содержимым файла xstartup. Форматирование переменной бу-

дет отличаться от того, что представлено в сценарии `xstartup`, однако редактирование оригинала не представляет трудности. К примеру, если необходимо вместо диспетчера `twm` вызывать оболочку GNOME, поменяйте `twm` на `gnome session`.

Внимание Перед редактированием сценария `vncserver` создайте его резервную копию. Если в результате изменений VNC-сервер не запустится можно вернуться к исходному варианту

Некоторые параметры, устанавливаемые в сценарии `vncserver`, перекрываются аргументами командной строки сценария. В частности, аргументы `-geometry` и `depth` позволяют задать размер рабочего стола и цветовую разрядность. Введите **`vncserver -help`** чтобы получить краткое описание параметров сценария

Установление VNC-соединения

Вызвать VNC-сервер во FreeBSD несложно. Для этого нужно ввести **`vncviewer`** на компьютере VNC-клиента, при необходимости указав имя сервера и через двоеточие — номер сеанса:

```
$ vncviewer bunyan.threeroomco.com:1
```

Эта команда подключает пользователя к первому VNC-сеансу на компьютере `bunyan.threeroomco.com`. Если опустить имя сервера и номер сеанса, утилита `vncviewer` запросит эту информацию. Если опустить только номер сеанса, будет подразумеваться сеанс 0. В любом случае утилита запрашивает пароль, созданный ранее с помощью команды `vncpasswd`. Есть несколько опций, модифицирующих работу утилиты `vncviewer`. Рассмотрим наиболее важные из них:

- **`-shared`** Обычно, когда пользователь подключается к VNC-сеансу, соединение предыдущего пользователя разрывается. Опция `-shared` запрещает серверу завершать исходное соединение. Это позволяет вести совместную работу на сервере. Можно, например, показать другому пользователю, как работать с той или иной программой. Мышь и клавиатура будут контролироваться обоими пользователями, что поначалу довольно непривычно.
- **`viewonly`**. Эта опция отключает мышь и клавиатуру в рамках сеанса. Можно лишь наблюдать функционирование существующего VNC-сеанса.
- **`-depth`**. Теоретически эта опция позволяет задать цветовую разрядность, но на практике такая возможность поддерживается редко.

После успешного подключения появляется окно, в котором отображается весь рабочий стол X Window System. К примеру, на рис. 21.4 представлено окно VNC-клиента, подключенного к VNC-серверу на другом компьютере (обратите внимание на рамку диспетчера окон вокруг рабочего стола).

VNC-клиенты для Windows и Mac OS имеют более сложные интерфейсы, но принцип их функционирования тот же: пользователь сообщает VNC-клиенту, к какому компьютеру и сеансу нужно подключиться, и вводит пароль, а программа открывает окно, в котором отображается рабочий стол сеанса. Если VNC-сервер работает в Windows или Mac OS, то им можно дистанционно управлять из FreeBSD, хотя это происходит медленнее, поскольку обработка системных вызовов осуществляется не так эффективно, как трансляция вызовов X Window System.

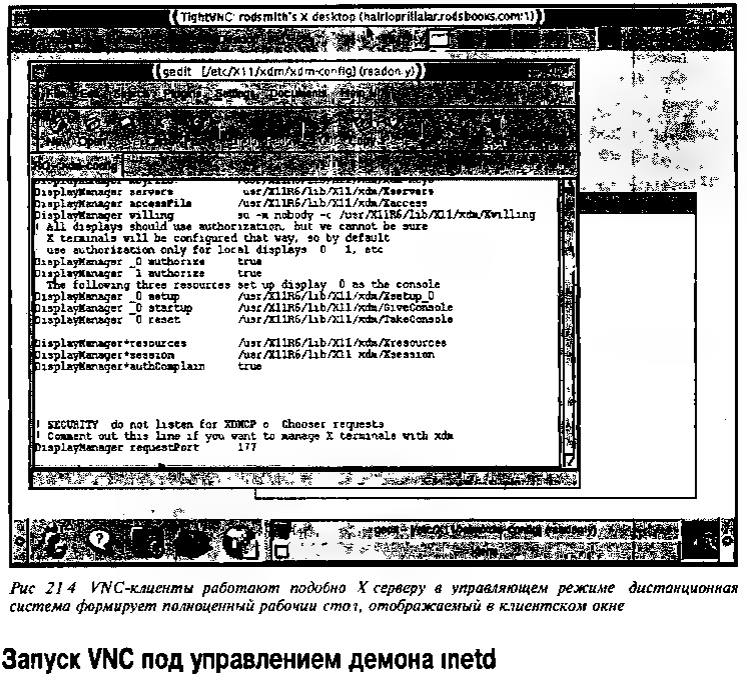


Рис 21 4 VNC-клиенты работают подобно X серверу в управляющем режиме дистанционная система формирует полноценный рабочии стол, отображаемый в клиентском окне

Запуск VNC под управлением демона inetd

В многопользовательской среде сразу проявляется неудобная особенность VNC каждый пользователь вынужден запускать сервер вручную и запоминать номер сеанса Во многих случаях гораздо проще было бы позволить VNC-серверу работать подобно большинству других серверов дистанционной регистрации, используя стандартную схему аутентификации пользователей FreeBSD Чтобы такая конфигурация стала возможной, выполните следующие действия

1 Добавьте в файл `/etc/services` запись для VNC-сервера Обычно этот сервер использует порт 5900 и несколько последующих портов Нужна всего одна запись

```
vnc      5900/tcp
```

2 Создайте для VNC-сервера запись в файле `/etc/inetd.conf`. В ней должна содержаться ссылка на программу `Xvnc` (реальный сервер, вызываемый сценарием `vncserver`):

```
vnc  stream  tcp  nowait  nobody  /usr/X11R6/b n/Xvnc  1 ↵
-inetd -query localhost -once
```

3 Перезапустите демон `inetd` с помощью команды **`killall -HUP inetd`**

4 Сконфигурируйте сервер XDMCP на компьютере VNC-сервера, как описывалось в главе 6 и в подразделе "Регистрация по протоколу XDMCP"

С этого момента VNC-сервер должен отвечать на запросы других компьютеров к сеансу 0, например **vncviewer bunyan:0** Если ввести такую команду или воспользоваться эквивалентными параметрами подключения в VNC-клиенте Windows или Mac OS, приглашение на ввод пароля не появится Вместо этого отобразится экран регистрации XDMCP, как если бы использовался XDMCP-клиент X-сервера Далее работа выполняется традиционным способом

В пунктах 1 и 2 следует обратить внимание на ряд чрезвычайно важных нюансов. Номер порта задаваемый в файле `/etc/services`, определяет номер сеанса для подключения к VNC-серверу. Порту 5900 соответствует сеанс 0, порту 5901 — сеанс 1 и т д Аргумент `:1` программы `Xvnc` в файле `inetd.conf` определяет номер X-сеанса соответствующего VNC-серверу Если X-сервер работает локально, то ему обычно сопоставлен сеанс 0 (`:1`) В противном случае аргумент можно опустить, и VNC-сервер попытается использовать сеанс 0. Опция `-inetd` заставляет VNC-сервер работать под управлением демона `inetd`, а опция `-once` заставляет VNC-сервер прекратить работу по окончании сеанса, что необходимо для демона `inetd` Опция `-query localhost` означает то же, что и аналогичная опция X-сервера· она подключает X-сервер к серверу XDMCP работающему на локальном компьютере.

Совет Если в рассматриваемой конфигурации VNC сервер работает не так как ожидается попробуйте ввести **`telnet vncserver 5900`**, где `vncserver` — имя компьютера VNC-сервера Указанная команда должна выдать сообщение об ошибке, которое поможет диагностировать проблему Например, если VNC-сервер пытается использовать X-сеанс, который уже кем-то занят утилита `telnet` не сможет подключиться к VNC-порту и сообщит об этом

Запуск VNC-сервера под управлением демона `inetd` приводит к отмене многих уникальных особенностей VNC

- VNC-сеансы больше не сохраняют свое состояние После завершения VNC сеанса придется регистрироваться заново, а если завершить сеанс, не выйдя из системы, все пользовательские процессы будут уничтожены, вследствие чего можно потерять данные Это также означает, что два пользователя не смогут подключиться к одному и тому же рабочему столу
- VNC-сервер принимает запросы на регистрацию с одинаковым номером сеанса
- Пользователям не нужно регистрироваться в системе, чтобы явно запустить VNC-сервер он всегда доступен
- Используются сценарии регистрации выполняемые сервером XDMCP а не файл `~/.vnc/xstartup`.
- Аутентификация осуществляется на основании имени и пароля пользователя FreeBSD, а не пароля пользователя VNC Негативным следствием этого факта является то, что пароль не шифруется

В целом запуск VNC сервера под управлением демона `inetd` удобен в многопользовательской среде, а также в том случае, когда требуется удаленный доступ, как к обычному X-серверу, но без сложностей, которые возникают при соответствующем конфигурировании Основная работа по конфигурированию переносится на компьютер VNC-сервера, а инсталляция и настройка VNC-клиентов очень проста В этом отличие от конфигурирования X-сервера, которое выполняется на обоих компьютерах

Конфигурацию демона `inetd` можно расширить, включив поддержку дополнительных VNC-сеансов. Повторите этапы 1–3, но присвойте службе новое имя (например, `vnc2`, а не `vnc`), назначьте ей другой номер порта (такой как 5901), задайте другой номер X-сеанса (например, `:2`) и передайте программе `Xvnc` необходимые

параметры в файле `inetd.conf` (например, `-geometry 700x500`). Это позволит VNC-серверу использовать разные установки в зависимости от потребностей клиентов. Пользователям придется запомнить, с каким номером VNC-сеанса связана та или иная конфигурация.

Резюме

В UNIX уже очень давно реализованы многопользовательские операции и средства дистанционного доступа. От предыдущих систем FreeBSD унаследовала текстовые средства дистанционного доступа, такие как Telnet и SSH, а также более поздние графические средства, в частности X Window System и VNC. Одни серверы можно сконфигурировать, поменяв всего одну-две строчки в конфигурационном файле. В случае других серверов требуется гораздо больше усилий, особенно при изменении установок по умолчанию. Но конечный результат оправдывает себя, ведь тем самым упрощается работа пользователей. Существенное внимание следует уделять безопасности серверов дистанционной регистрации, поскольку эти серверы почти не ограничивают действия пользователей, а кроме того, при регистрации на сервере по сети передаются пароли. Таким образом, по возможности рекомендуется применять механизмы шифрования паролей, например SSH.

Полный справочник по

Глава 22

Вспомогательные серверы

В предыдущих главах рассматривалось конфигурирование ключевых типов серверов, таких как почтовые и Web-серверы. Существует также целый ряд других, не менее важных серверов, с которыми реже приходится иметь дело пользователям. В этой главе описываются следующие серверы:

- **Серверы DHCP** — предоставляют другим компьютерам информацию о конфигурации сети.
- **Серверы DNS** — преобразуют имена компьютеров в IP-адреса и наоборот.
- **Серверы синхронизации времени** — позволяют всем компьютерам сети настраивать системные часы по единому эталону.
- **Серверы шрифтов** — предоставляют X-серверам согласованный набор шрифтов.

Серверы DHCP и DNS обеспечивают функционирование сети Internet и множества подсетей. Поскольку они очень сложны, мы познакомимся лишь с основами их конфигурирования. В этом смысле другие два типа серверов намного проще. Серверы синхронизации времени важны для систем типа Kerberos (`http://web.mit.edu/kerberos/www/`), функционирование которых зависит от синхронной работы часов двух компьютеров.

Серверы DHCP

DHCP (Dynamic Host Configuration Protocol - протокол динамического конфигурирования компьютеров) является популярным протоколом присвоения компьютерам IP-адресов и передачи им сопутствующей информации о настройках сети. Применение протокола позволяет упростить конфигурацию компьютеров в сети (в главе 14 рассказывалось о настройке DHCP-клиентов). Платой за такое упрощение является необходимость наличия DHCP-сервера. Во FreeBSD есть популярный DHCP-сервер dhcpd, который прекрасно работает в этой системе. Ниже будут приводиться аргументы за и против запуска DHCP-сервера, после чего мы поговорим о том, как сконфигурировать сервер на предоставление динамических и статических IP-адресов.

Когда запускать DHCP-сервер

Чтобы ответить на вопрос, запускать или не запускать DHCP-сервер, нужно оценить, приведут ли затраты на конфигурирование сервера к эквивалентному сокращению затрат на конфигурирование DHCP-клиентов. Из главы 14 читатели могли составить представление о том, какие усилия требуется прикладывать на стороне клиента. Естественно, в системах, отличных от FreeBSD, действия по конфигурированию будут другими, но объем затрат примерно такой же. Конфигурировать DHCP-сервер сложнее, чем настраивать одиночный компьютер на использование статического IP-адреса. Как правило, затраты начинают окупаться при количестве компьютеров от 6 до 12. На оценку влияет несколько факторов:

- **Потребность в статических IP-адресах.** Статическая конфигурация, при которой выделяемый IP-адрес гарантированно не меняется, — более трудоемкая, чем в случае динамического выделения адресов. Таким образом, если в сети функционирует много серверов, которым нужны статические адреса, применение DHCP может не привести к упрощению процедуры администрирования.
- **Мультисистемные и переменные конфигурации.** Если в сети много компьютеров, загружающихся в мультисистемном режиме или используемых для тестирова-

ния операционных систем, то протокол DHCP особенно удобен, поскольку он упрощает конфигурирование каждой операционной системы компьютера

- **Портативные компьютеры.** Если в сети есть компьютеры, подключенные не постоянно, например ноутбуки, то применение протокола DHCP позволяет существенно упростить их конфигурирование. Это особенно справедливо в том случае, когда ноутбук регулярно подключается к нескольким сетям, в каждой из которых используется DHCP.
- **Системы, конфигурируемые пользователями** Если пользователи самостоятельно конфигурируют свои компьютеры применение DHCP поможет снизить вероятность ошибок
- **Слишком большое число компьютеров** Когда количество компьютеров превышает число доступных IP-адресов, но не все компьютеры постоянно функционируют, без DHCP не обойтись. DHCP-сервер выделит компьютеру IP-адрес во временное пользование, и по завершении работы компьютера этот адрес снова станет доступным В такой среде необходимо задавать короткий период аренды адресов (об этом рассказывается на с. 432)

В главе 17 описывался NAT (Network Address Translation) другой механизм эффективного использования ограниченного числа IP-адресов. Благодаря NAT можно задействовать больше компьютеров, чем позволяет имеющийся диапазон IP-адресов, даже если все эти компьютеры будут функционировать одновременно.

В целом DHCP лучше всего использовать в сетях с большим количеством компьютеров и в сетях, число действующих компьютеров в которых часто меняется В очень маленьких сетях а также в ситуациях, когда большинству компьютеров требуются статические IP-адреса, польза от DHCP будет невелика

В стандартной конфигурации DHCP-сервер реагирует на *широковещательные запросы* от клиентов, т е на пакеты, адресованные всем компьютерам сети Такая схема работы принята потому, что на момент начальной загрузки DHCP-клиенту не известен ни собственный IP адрес, ни адрес DHCP-сервера Широковещательный запрос обязательно попадет к DHCP-серверу, если тот находится в локальной подсети В очень большой сети, подсети которой разделены маршрутизаторами, нужно либо запускать DHCP-сервер в каждой подсети, либо специальным образом конфигурировать маршрутизаторы, чтобы они ретранслировали такого рода запросы между подсетями

Инсталляция DHCP-сервера

Во FreeBSD чаще всего используется DHCP-сервер ISC (Internet Software Consortium консорциум разработчиков программного обеспечения для Internet) Он доступен как в виде пакета, так и через коллекцию переносимых приложений (`isc-dhcpd`)

Конфигурационные файлы сервера находятся в каталоге `/usr/local/etc` Наиболее важные из них перечислены ниже

- `/usr/local/etc/rc.d/isc-dhcpd sh` — сценарий, управляющий запуском сервера. Вместе с пакетом инсталлируется сценарий `isc-dhcpd.sh.sample`, который достаточно переименовать, чтобы сервер запускался автоматически Редактировать этот файл обычно не приходится
- `/usr/local/etc/dhcpd.conf` — основной конфигурационный файл DHCP сервера Его структура будет описана далее. Как и в случае сценария запуска, сразу после инсталляции этот файл имеет дополнительное расширение `sample` Скопируйте или переименуйте его чтобы DHCP-сервер стал доступен

- **/usr/local/etc/rc.isc-dhcpd conf** — сценарий с параметрами запуска передаваемыми серверу Обычно редактировать его не приходится но если нс обходимо, чтобы сервер прослушивал другой набор сетевых интерфейсов ука жите их список в переменной dhcpd_ifaces.

Файл dhcpd.conf содержит комментарии (строки начинающиеся с символа '#'), глобальные параметры (после названия параметра указывается одно или несколько значений) и определения подсетей, которые представляют собой наборы параметров, применяемые к конкретным подсетям Определения параметров заканчиваются символом ';', а блоки параметров заключаются в фигурные скобки например:

```
subnet 10.254.239 0 netmask 255.255.255.224 {
   range 10.254.239 10 10.254.239.20;
   option routers rtr-239 0-1.example org; rtr 239 0 2 example org,
}
```

Прежде чем запускать DHCP-сервер создайте файл, в котором будет храниться информация об арендуемых ресурсах Для этого достаточно ввести **touch /var/db/dhcpd leases** В случае отсутствия файла сервер не сможет нормально функционировать

Основы конфигурирования

Файл dhcpd conf.sample, поставляемый вместе с FreeBSD вполне функционален, но его почти наверняка придется настраивать в соответствии с потребностями конкретной сети Некоторые из глобальных параметров содержат вполне разумные значения по умолчанию, другие необходимо менять. В демонстрационный файл включено также несколько определений подсетей, которые нужно заменить собственными определениями.

На заметку DHCP сервер предоставляет IP-адреса *другим* компьютерам Самому серверу нужно назначить статический IP адрес

Задание глобальных параметров

Глобальные параметры стоят в начале файла dhcpd conf Рекомендуется взять за образец файл dhcpd.conf.sample Большинство заданных в нем глобальных параметров можно оставить без изменений. Ниже перечислены параметры на которые следует обратить первоочередное внимание.

- **option domain-name** "*имя_домена* — задает имя домена передаваемое клиентам
- **option domain-name-servers** *список_серверов* — позволяет сообщать клиентам, какие компьютеры являются DNS-серверами. Список серверов разделяется запятыми. В нем можно указывать как IP-адреса, так и имена компьютеров. Во втором случае демон dhcpd самостоятельно преобразует их и возвращает полученные IP-адреса клиентам.
- **option routers** *список_маршрутизаторов* — позволяет сообщать клиентам IP-адреса маршрутизаторов (шлюзов) В большинстве случаев в списке присутствует один-единственный адрес, но в сложных сетях ситуация может быть иной. Вместо IP-адреса разрешается указывать доменное имя
- **default-lease-time** *секунды*. Функционирование протокола DHCP основано на принципе *аренды* Срок аренды ограничен когда он истекает DHCP-сервер волен переприсвоить IP-адрес другому компьютеру. Оптимальный период аренды зависит от конфигурации сети Если число подключенных к сети

компьютеров часто меняется, период аренды должен быть непродолжитель ным, например 3600 (т.е. один час). Если же состояние сети меняется нечасто, то удлинение срока аренды до недели (и даже больше) позволяет сократить DHCP-трафик и свести к минимуму риск вследствие сбоя DHCP-сервера. Значение по умолчанию установленное в демонстрационном файле, — 600, т.е. 10 минут. Такая продолжительность оправдана при тестировании или отладке, но для нормальной работы сети она недостаточна

- **max-lease-time** *секунды* — задает максимальную продолжительность арен ды, принимаемую сервером в том случае, если клиент запрашивает увеличение срока аренды

В более сложных конфигурациях изменению могут подвергаться и другие установ ки, например параметры, определяющие взаимодействие DHCP-сервера с DNS сервером или доставку информации о клиентах NetBIOS

Определение подсети

Наиболее короткое определение подсети выглядит так

```
subnet 10.152 187.0 netmask 255.255.255.0 {
}
```

В начальной строке указаны адрес подсети (10.152 187.0) и сетевая маска (255.255 255 0). В данном примере объявлена подсеть, *не* обслуживаемая сервером, — об этом свидетельствует пустой блок параметров. Такого рода определение означает что компьютер DHCP-сервера включен в подсеть, но не выполняет для нее функции собственно DHCP-сервера В качестве примера можно привести маршрутизатор реа лизующий DHCP-функции только в одной подсети хотя из соображений безопасно сти не рекомендуется запускать DHCP-сервер на маршрутизаторе В большинстве случаев определение подсети содержит хотя бы одну дополнительную строку

```
subnet 10.152.187.0 netmask 255.255.255 0 {
   range 172.29.30.20 172.29.30.254;
}
```

Ключевое слово `range` задает диапазон IP адресов предоставляемых клиентам В этом простом примере DHCP-сервер выделяет адреса любым компьютерам при усло вии, что список доступных адресов еще не исчерпан

Не включите по неосторожности собственный IP-адрес DHCP-сервера в список адресов, предоставляемых клиентам. Дублирование адресов в пределах одной подсети приведет к тому что как минимум одна из систем окажется недоступной.

В определениях подсетей могут присутствовать упоминавшиеся ранее параметры К примеру, допускается переопределять стандартную продолжительность срока аренды

Назначение статических IP-адресов

В базовой конфигурации DHCP-сервера IP-адреса назначаются случайным обра зом Клиенты принимают те адреса, которые им предоставляет сераер. Если пользователь ненадолго выключит компьютер, то, вернувшись через несколько минут, он может обнаружить, что у компьютера уже другой IP-адрес Вследствие этого на компьютерах DHCP-клиентов сложно запускать серверные программы ведь другим компьютерам непросто узнать IP-адрес такого сервера Наиболее простой выход из ситуации — заставить DHCP-сервер назначать конкретному компьютеру один и тот же IP-адрес в процессе загрузки Для этого в конфигурационном файле DHCP-сервера нужно создать специальные клиентские записи

На заметку

IP адреса как правило изменяются при перезагрузке компьютера или активизации сетевого интерфейса Маловероятно чтобы адрес изменился в процессе работы компьютера. Когда заканчивается половина срока аренды, клиент просит DHCP-сервер продлить аренду Сервер, если не происходит ничего экстраординарного, удовлетворяет этот запрос и продлевает аренду IP-адреса Таким образом, если DHCP-клиент работает без сбоев, его IP-адрес остается неизменным

Статический IP-адрес закрепляется за MAC-адресом сетевого интерфейса клиентского компьютера Соответствующее объявление выглядит так (оно может встречаться как внутри определения подсети так и вне его).

```
host teela {
   hardware ethernet 00:05:02:a7:76:da;
   fixed-address 192.168 1.2;
```

После ключевого слова `host` указано имя искомого компьютера (без доменного расширения), а затем – блок параметров. Параметр `hardware ethernet` задает MAC-адрес платы Ethernet (существует аналогичный параметр `hardware token-ring` для плат Token Ring). Параметр `fixed-address` задает IP-адрес, связываемый с этим MAC-адресом. Когда DHCP-сервер получает запрос от компьютера с MAC-адресом 00.05·02 A7:76.DA, то выделяет ему IP-адрес 192.168 1 2. Сераер также сообщает клиенту его имя (в данном случае `teela`), но многие клиенты его нгнорируют Никакого специального конфигурирования клиента не требуется

Узнать MAC-адрес клиента можно по-разному. Во FreeBSD к примеру после загрузки системы можно ввести такую команду'

```
$ dmesg | grep "Ethernet address"
fxp0: Ethernet address 00:03:47·b1:ee:b8
```

Информацию о MAC-адресах можно запрашивать и с DHCP-сервера. Один из вариантов — воспользоваться командой `arp`, которая позволяет работать с низкоуровневыми таблицами IP- и MAC-адресов Приведенная ниже команда сообщает MAC-адрес заданного компьютера, при условии, что его IP-адрес известен

```
$ arp 192.168.1 2
```

На заметку

Иногда, чтобы эта команда заработала, нужно предварительно обратиться к компьютеру, например послать ему команду `ping`.

Рассмотренный метод неудобен тем, что сервер должен заранее знать IP-адрес клиента Можно временно прибегнуть к динамическому назначению адресов или вручную присвоить компьютеру временный статический IP-адрес, с тем чтобы потом перезагрузить компьютер и заставить его выбрать необходимый адрес. Другой вариант — посмотреть надписи непосредственно на сетевой плате, где иногда указывается MAC-адрес

Серверы DNS

DHCP-серверы незаметно функционируют во многих сетях проявляя себя лишь в случае сбоя То же самое можно сказать и о серверах DNS (Domain Name System — система доменных имен). Как рассказывалось в главе 14, они преобразуют имена компьютеров в IP-адреса и наоборот (такое преобразование часто называют *распознаванием*) В небольших сетях и даже в сетях среднего размера DNS-серверы могут работать на обычных компьютерах, не предъявляя особых требований к их производн

тельности Часто они функционируют на том же оборудовании что и DHCP-серверы Помимо сценария запуска в конфигурировании сервера участвует еще несколько фай лов Один из них хранит общие параметры сервера, а остальные файлы содержат ин формацию о доменах, обслуживаемых сервером

Когда запускать DNS-сервер

DNS-серверы предоставляют услуги пользователям двух категорий.

- **Внутренние пользователи** Компьютеры локальной сети посылают запросы ло кальному DNS-серверу для распознавания доменных имен или IP-адресов В ответ DNS сервер сообщает IP-адреса, связанные с заданными доменными именами или обращается к внешним DNS-серверам чтобы получить у них информацию о неизвестном ему доменном имени
- **Внешние пользователи** Администраторы, обслуживающие собственные домены должны запустить как минимум два DNS-сервера для преобразования имен в рамках домена На практике для этой цели предпочтительнее подключать внешние службы, особенно в случае небольших доменов Правда, если в доме не используются статические IP-адреса, непосредственно доступные из внеш них доменов, то можно запустить и собственные DNS-серверы. Это позволит администратору самому контролировать все аспекты работы серверов В более сложных конфигурациях можно интегрировать DNS-сервер с DHCP-сервером, чтобы они динамически обновляли записи в базах данных при изменении IP адресов

Как правило, запускать собственный DNS-сервер имеет смысл только тогда, когда число компьютеров в сети переваливает за несколько десятков, причем эти компьюте ры должны ссылаться друг на друга по именам Если же в сети один-два сервера и много клиентов, то DNS-сервер не нужен Вместо этого достаточно задать в клиент ских файлах `/etc/hosts` (или их аналогах в других операционных системах, напри мер `C:\WINNT\system32\drivers\etc\hosts` в Windows 2000) ссылки на серверы.

На заметку

В некоторых протоколах используются собственные механизмы распо знавания имен В качестве примера назовем NetBIOS на основе кото рого функционируют механизмы совместного доступа к файлам в Windows и программа Samba

Наиболее распространенный DNS-сервер во FreeBSD — пакет BIND (Berkeley Internet Name Domain — система доменных Internet-имен реализации университета Беркли) Серверный демон называется named, он инсталлируется во FreeBSD по умолчанию При желании, конечно, можно воспользоваться и другими серверами, та кими как `djbdns` и `dns_balance` (первый из них доступен только через коллекцию переносимых приложений), но, учитывая огромную популярность BIND, именно его конфигурирование мы и рассмотрим

Внимание

В этой главе описывается базовое конфигурирование сервера BIND которое уместно в домашних и небольших коммерческих сетях. В бо лее сложных случаях следует обращаться к документации и специаль ной литературе Ошибка в конфигурации сервера вызвать серьезные проблемы в работе домена

Чтобы организовать запуск сервера BIND отредактируйте файл `/etc/rc conf` включив в него следующую строку

```
named_enable="YES"
```

Наличие этой строки приведет к вызову сервера при очередной перезагрузке компьютера. Для однократного запуска сервера достаточно ввести **named** в режиме суперпользователя. Если в конфигурацию сервера вносились изменения, то для обновления конфигурации нужно ввести **killall -HUP named**, опять же от имени суперпользователя. Иногда приходится пользоваться командой **killall named**; **named** она останавливает и перезапускает сервер, полностью очищая его кэш-буфер.

Базовое конфигурироввние серверв

Основной конфигурационный файл сервера BIND называется `/etc/namedb/named.conf`. В нем задаются глобальные параметры и содержатся ссылки на дополнительные файлы, определяющие конфигурацию конкретных *зон* (доменов или групп сетей, обслуживаемых сервером BIND). Формат этих файлов будет описан далее.

Строки файла `named.conf`, начинающиеся с символов '//' или '#', считаются комментариями. Допускаются многострочные комментарии в стиле языка C: начало комментария обозначается символами `/*`, а конец — `*/`. Остальные строки разбиты на группы, напоминающие определения подсетей в конфигурационном файле DHCP-сервера: только после закрывающей фигурной скобки ставится точка с запятой (';'). Наиболее важны следующие группы параметров: `option` и `zone`. В первой задаются общие параметры сервера, вторая содержит ссылку на файл конфигурации конкретной зоны. В листинге 22.1 приведен пример конфигурационного файла сервера BIND. В стандартном файле, входящем в состав FreeBSD, содержатся многочисленные комментарии, поясняющие назначение различных компонентов и задается ряд дополнительных установок, отсутствующих в листинге.

Листинг 22.1. Образец файла `/etc/named/named.conf`

```
options {
    directory "/etc/namedb";
    pid-file "/var/run/named/pid"
    forward first;
    forwarders {
       10.202.45.108;
       172.20.232.1;
    };
    listen-on {
       192.168.1.4
    }
};
zone "." {
    type hint;
    file "named.root"
};
zone "threeroomco.com" {
    type master;
    file "named.threeroomco.com"
};
zone "1.168.192.in-addr.arpa" {
    type master;
    file "localhost.rev";
}
zone "0.0.127.in-addr.arpa" {
    type master;
    file "named.local";
}
```

Рассмотрим основные компоненты этой конфигурации.

- **Параметр `directory`** — задает каталог в котором находятся файлы зон В стандартной инсталляции FreeBSD это тот же каталог в котором находится файл `named.conf`
- **Переадресация запросов** Стандартная схема работы DNS-сервера такова· чтобы получить информацию о доменном имени. сервер опрашивает целый ряд других DNS-серверов, начиная с *корневых*. Корневым серверам известны адреса серверов, которые владеют информацией о компьютерах доменов верхнего уровня, таких как `.com` и `.ru`. Такой режим называется *полным рекурсивным поиском*. В качестве альтернативы можно включить режим *переадресации*, когда сервер посылает запрос только одному внешнему DNS-серверу, который выполняет полный рекурсивный поиск. Переадресация имеет смысл в небольших сетях с невысокой скоростью подключения к Internet. Параметр `forward first` заставляет сервер BIND выполнять переадресацию поиска в первую очередь а если она потерпит неудачу, переходить к полному рекурсивному поиску Другой вариант — параметр `forward only`, заставляющий сервер выполнять только переадресацию. Именно он установлен по умолчанию во FreeBSD. В обоих случаях в списке `forwarders` указываются адреса внешних DNS-серверов, опрашиваемых в режиме переадресации
- **Привязка к интерфейсам**. Параметр `listen-on` заставляет сервер прослушивать сетевые интерфейсы связанные с заданными IP-адресами. Этот параметр используется в том случае если у компьютера несколько сетевых интерфейсов но отвечать на запросы нужно только по одному из них
- **Определения зон** В листинге 22 1 содержится четыре определения зоны. Первая из них (определена как `zone "."`) — это *корневая зона*. В нее входят все адреса сети Internet, за исключением тех, что описаны в более специализированных файлах зон Информацию об этой зоне менять не нужно Строка `type hint` заставляет сервер BIND выполнять полный рекурсивный поиск или переадресацию поиска для адресов этой зоны В остальных зонах определяются домены (`threeroomco.com`) или блоки адресов (192 168 1 0/24 и 127 0 0.0/24). IP-адреса записываются в обратном порядке с добавлением суффикса `in addr.arpa`, например `1.168.192.in-addr.arpa` вместо 192 168 1.0/24 (только сетевая часть адреса преобразуется в имя зоны) Для всех зон параметр `file` задает имя файла, содержащего описания имен и адресов зоны

Важно понимать разницу между зонами прямого и обратного преобразования. Рассмотрим небольшую коммерческую сеть с десятком компьютеров, подключенную к Internet Компания управляет собственным доменом (допустим, `threeroomco.com`) и потому должна поддерживать возможность прямого поиска. Это значит, что, когда пользователи запрашивают IP адрес компьютера `www.threeroomco.com`, сервер компании должен возвращать соответствующее значение (172.17.202.7) В то же время сами IP-адреса находятся под контролем провайдера компании. Сервер провайдера управляет зоной обратного преобразования (`202.17.172.in-addr.arpa`), информацией о которой не владеет сервер компании. Противоположное будет справедливо лишь в том случае, если компания получит фиксированный блок IP-адресов в постоянное пользование и договорится с провайдером о подключении этих адресов к Internet. Схема распределения зон, представленная в листинге 22 1, подходит для DNS-сервера небольшой сети особенно если это частная сеть (скрыта за маршрутизатором NAT).

В случае нескольких DNS-серверов один из них является *подчиненным* а другой *главным* В такой конфигурации подчиненный сервер периодически синхронизирует содержимое файлов зон с главным сервером, запрашивая полную информацию о зонах. Вот для чего нужны строки `type master` в определениях зон Они объявляют

сервер главным для соответствующей зоны. Конфигурация подчиненного сервера аналогична, только в определениях зон присутствует строка type slave после которой определяется главный сервер

```
masters {
   192.168 1 1
}
```

Конфигурироввние зоны

Параметры зоны содержатся в файле зоны имя которого задается в директиве zone файла named.conf. Описания зон прямого и обратного преобразований имеют много общего но есть и важные различия, о которых будет рассказано ниже

На заметку Если вы договариваетесь с провайдером о предоставлении DNS услуг для своего домена, то должны предоставить ему и соответствующую информацию о зонах

Настройка прямого преобразования

Файл зоны прямого преобразования состонт из набора записей, описывающих домен В первой записи указаны характеристики самой зоны, например то, как долго другие DNS-серверы должны хранить информацию о зоне в своих кэш-буферах. В последующих записях описываются отдельные имена в пределах домена. Комментарии помечаются символом ';' и могут стоять как в начале, так и в конце строки В листинге 22.2 показан небольшой, но вполне функциональный пример файла зоны.

Листинг 22.2. Пример зоны прямого преобразования

```
threeroomco com    IN  SOA  zeus.threeroomco.com. \
                            admin threeroomco.com. (
                       2002101003 , порядковый номер
                       28800      , период обновления
                       14400      ; интервал между попытками
                       3600000    , интервал устаревания
                       86400      · стандартное время жизни

mail                   IN  A      172.20 29.101
zeus                   IN  A      172.20.29.78
jupiter                IN  CNAME  zeus
@                      IN  MX     10  mail
threeroomco com        IN  MX     20  mail.abigisp.net.
@                      IN  NS     zeus threeroomco.com.
```

Рассмотрим структуру записей файла

- **Имя узла** В первом поле указывается имя компьютера или домена, например mail или zeus. В самой первой строке задано имя домена, соответствующего зоне В конце полных доменных имен стоит точка. Имена без точки воспринимаются как имена компьютеров в рамках основного домена. К примеру, имя mail раскрывается как mail threeroomco.com. Знак '@', встречающийся в двух записях, означает "доменное имя, указанное в первой записи"
- IN Второе поле равно IN, что означает сеть Internet. Это стандартный компонент всех записей
- **Тип записи.** Тип записей указывается после поля IN Первая запись листинга имеет тип SOA (start of authority — начало передачи полномочий). Это главная запись зоны, определяющая ее характеристики в целом. Запись типа A (address) связывает IP адрес с именем а запись типа CNAME (canonical

name — каноническое имя) связывает синонимичные имена. В листинге 22.2 имя jupiter определено как синоним имени zeus, т.е. к одному и тому же компьютеру можно обратиться по двум именам. Записи типа MX (mail exchange — обмен почтой) сообщают внешним почтовым серверам о том, какие компьютеры используются для доставки почты, адресованной домену. На конец, записи типа NS (name server — сервер имен) ссылаются на серверы имен зоны. В первом поле записей MX и NS должно быть указано имя домена. Это можно сделать явно или с помощью символа '@'.

- **Параметры записи.** Записи CNAME и A содержат единственный элемент: имя или IP-адрес. В записи MX указаны имя или IP-адрес почтового сервера и значение его приоритета. Внешние почтовые серверы, доставляющие почту в домен, опрашивают MX-системы в порядке приоритета, начиная с наименьшего значения.

Запись SOA является наиболее сложной из всех и содержит несколько информационных элементов.

- **Сервер имен.** В записи SOA указывается имя самого DNS-сервера, в данном случае zeus.threeroomco.com. (обратите внимание на точку в конце).
- **Почтовый адрес администратора.** После имени DNS-сервера указан почтовый адрес администратора зоны. При отправке почты замените первую точку знаком '@' и удалите точку в конце (admin@threeroomco.com).
- **Параметры обновления данных.** Числа в скобках касаются устаревания данных о зоне в кэш-буферах других DNS-серверов и определяют периодичность, с которой подчиненные серверы должны запрашивать у главных серверов обновленные данные. Каждое значение в листинге 22.2 сопровождается комментарием. Порядковый номер позволяет подчиненным серверам узнавать об изменениях файлов зон. Его нужно изменять всякий раз при внесении модификаций в файл. Часто в порядковом номере кодируется дата в формате *ггггммдд* с добавлением более короткого номера. Остальные значения определяют интервалы времени в секундах. Период обновления говорит о том, как часто нужно обновлять данные. Следующее значение задает промежуток времени перед повторной попыткой на случай, если запрос завершился неудачей. Интервал устаревания определяет, сколько времени подчиненный сервер должен хранить информацию о зоне в случае недоступности главного сервера. Последнее значение определяет стандартное время жизни (TTL — time to live) кэшированных записей. Если в конфигурацию DNS часто вносятся изменения, значение TTL должно быть небольшим (несколько часов).

Файл, представленный в листинге 22.2, можно взять за основу для небольших доменов. Многие детали, конечно, придется поменять, но записи других типов скорее всего не понадобятся.

Настройка обратного преобразования

В листинге 22.3 показан файл зоны обратного преобразования, дополняющий листинг 22.2. В целом они похожи, но есть важные отличия. Этот файл тоже начинается с записи SOA, однако имя зоны относится к псевдодомену in-addr.arpa, охватывающему IP-адреса. Когда компьютеру нужно узнать доменное имя, соответствующее заданному IP-адресу, он меняет порядок элементов адреса на противоположный, добавляет к нему суффикс .in-addr.arpa и посылает запрос, как будто для прямого преобразования. В результате он получает искомое доменное имя.

Вместо записей A и CNAME в файлах обратного преобразования используются записи PTR (pointer — указатель), связывающие IP-адреса с доменными именами, при

чем каждому адресу ставится в соответствие одно имя. Важно указывать полное имя, включая точку, иначе сервер BIND будет считать, что имя относится к псевдодомену `in-addr.arpa`. Разрешается сокращать IP-адреса, как в случае с адресом 172.20.29.78 в листинге 22.3 (последняя строка).

Листинг 22.3. Пример зоны обратного преобразования

```
29.20.172.in-addr.arpa.    IN  SOA  zeus.threeroomco.com. \
                                    admin.threeroomco.com  (
                          2002101003 ; порядковый номер
                          28800      ; период обновления
                          14400      ; интервал между попытками
                          3600000    ; интервал устаревания
                          86400      ; стандартное время жизни
                         )
@                              IN  NS   zeus.threeroomco.com.
101.29.20.172.in-addr.arpa     IN  PTR  mail.threeroomco.com.
78                             IN  PTR  zeus.threeroomco.com
```

Помните, что зоны прямого и обратного преобразований могут не охватывать одни и те же компьютеры. К примеру, у провайдера могут быть пользователи, которым нужно, чтобы IP-адреса преобразовывались в имена их собственных доменов, а не доменов провайдера. Тогда в файле зоны обратного преобразования будут присутствовать записи PTR, ссылающиеся на множество различных доменов. В свою очередь, в клиентских файлах зон прямого преобразования могут содержаться записи, ссылающиеся на IP-адреса в различных сетях (офисы в других городах или серверы Web-хостинга).

Серверы синхронизации времени

Как гласит поговорка, человек с часами знает время, а человек с двумя часами не уверен в том, что оно точное. У каждого компьютера есть системные часы, поэтому в сетевой среде расхождения во времени неизбежны. Хуже того, некоторые протоколы, например Kerberos, требуют, чтобы часы компьютеров были синхронизированы. Клиентские утилиты могут дать сбой, если окажется, что метка времени, проставленная файловым сервером, не соответствует их локальному времени. Когда составляется отчет о взломе системы, важно иметь под рукой журнальные файлы с метками времени, но их ценность может резко уменьшиться, если окажется, что системные часы отстают, пусть даже на несколько секунд. Вот почему так важно синхронизировать часы компьютеров между собой, а в идеале — с внешним эталонным источником.

Существует несколько средств синхронизации системных часов. Одно из наиболее популярных и точных из них — NTP (Network Time Protocol — протокол сетевой синхронизации времени). Он позволяет синхронизировать часы двух компьютеров с точностью до секунды, к тому же в Internet существует целый ряд NTP-серверов, позволяющих сверять системные часы с внешним источником.

Протокол NTP

В основе протокола NTP лежит использование источника точного времени. Это могут быть атомные часы или часы, синхронизируемые по сигналам GPS (Global Positioning System — глобальная система позиционирования). Такого рода источник называется часами *уровня 0* (stratum 0). Прямой доступ к подобному источнику времени можно получить только с помощью специального оборудования, но обычный компьютер может синхронизировать свои часы с уровнем 0. Такой компьютер называется

источником времени *уровня 1*. Серверы уровня 1 являются наиболее точными источниками времени в Internet, по крайней мере теоретически (все упирается в качество часов уровня 0). Они обслуживают серверы уровня 2 и т.д. Далее будет рассказано о том, как обнаружить сервер времени, по которому можно синхронизировать часы компьютера. Как правило, в сети работает локальный сервер времени, обслуживающий все компьютеры сети. Это резко снижает нагрузку на внешние серверы синхронизации.

На каждом уровне иерархии серверов настройка осуществляется путем обмена пакетами с информацией о времени. Проблема заключается в том, что пакеты не доставляются мгновенно, особенно если родительский сервер находится в Internet. Отсюда возникают различные задержки. Вот почему нельзя просто запросить время у сервера и установить его на компьютере. Точность такого времени оценить сложно. Сервер NTP компенсирует задержки, определяя, сколько времени шел ответ от внешнего сервера, и вычисляя среднее время передачи пакетов. В результате точность системных часов будет составлять несколько миллисекунд, при условии, что сеть не слишком перегружена.

Во FreeBSD протокол NTP реализован в демоне `ntpd` (более ранняя версия называлась `xntpd`). Этот демон является одновременно и клиентом сервера более низкого уровня (родительского), и сервером для компьютеров более высокого уровня. Нередко на одном из компьютеров сети демон `ntpd` выполняет обе функции, а остальные компьютеры используют демон лишь для синхронизации с первым компьютером. Возникает вопрос: зачем использовать полноценный сервер, если нужен обычный клиент? Причина в том, что демон способен проверять время на регулярной основе, обеспечивая равномерный ход часов. В качестве альтернативы можно запускать NTP-клиент средствами демона `cron` или просто на этапе начальной загрузки системы. Существуют программы, функционирующие только как NTP-клиенты, но гораздо проще использовать демон `ntpd`.

Поиск родительских серверов

NTP-серверы упорядочены в иерархию, основанную на расстоянии от источника точного времени. Идеальным вариантом была бы синхронизация с сервером самого низкого доступного уровня, 1. Но если бы все так поступали, серверы уровня 1, имеющиеся в Internet, быстро переполнились бы запросами, что вызвало бы коллапс всей системы. Поэтому на практике основные серверы небольших сетей лучше синхронизировать с серверами уровня 2. Снижением точности вполне можно пренебречь, так как гораздо важнее равномерно распределить нагрузку на серверы. Граница между "небольшой" и "крупной" сетью проходит примерно на отметке 100 компьютеров.

Чтобы найти подходящий сервер уровня 2, обратитесь к ресурсу `http://www.eecis.udel.edu/~mills/ntp/clock2a.htm`, где представлен список доступных серверов (на момент написания книги их насчитывалось 157). Найдите сервер, который расположен поблизости, и запишите его доменное имя и IP-адрес. Многие организации требуют, чтобы пользователи связывались с ними, прежде чем осуществлять синхронизацию с их помощью. Если для выбранного сервера это актуально, выполните просьбу его операторов.

Поскольку ошибки измерений в протоколе NTP связаны с задержками пакетов, время реакции родительского сервера должно быть минимальным. Выберите несколько потенциальных кандидатов и отправьте им `ping`-запросы, чтобы определить, какой сервер реагирует быстрее.

Помимо общедоступных серверов уровня 2 существуют и другие серверы. К примеру, многие провайдеры предлагают услуги точного времени для своих клиентов. Их

серверы обычно расположены ближе, поэтому их выбор более предпочтителен с точки зрения продолжительности задержек

К сожалению, NTP-серверы не всегда доступны Сетевой канал может выйти из строя, сервер может прекратить работу Хуже всего, когда сервер синхронизируется по ненадежному источнику Чтобы избежать подобных проблем, основной NTP-сервер сети часто синхронизируют с несколькими внешними NTP-серверами Трех серверов обычно достаточно. Демон `ntpd` способен диагностировать некачественные серверы и автоматически игнорировать их синхросигналы. Впрочем, для небольшой сети использование внешних NTP-серверов чересчур накладно. Если проблема носит временный характер, демон `ntpd` вполне может обеспечить точный ход часов, поскольку он отслеживает отклонение от нормального хода и способен его компенсировать даже в отсутствие источника точного времени.

Независимо от того, как синхронизируется главный сервер системные часы всех остальных компьютеров сети должны быть синхронизированы с главным сервером Если в сети 50 компьютеров, это позволит весьма существенно снизить нагрузку на внешние серверы без заметного ущерба для точности В крупных организациях можно запустить два или даже больше локальных серверов в целях избыточности и синхронизировать часы по всем локальным серверам

Конфигурирование протокола NTP

Во FreeBSD есть два средства синхронизации. Первое из них — это NTP-клиент `ntpdate`, устанавливающий часы компьютера в ходе загрузки системы Второе — демон `ntpd`, который непрерывно сверяет и настраивает показания часов и сообщает время другим компьютерам

Настройка таймера на этапе начальной загрузки

Если система часто перезагружается, можно настраивать ее часы на этапе начальной загрузки, что обеспечит приемлемую точность Включить эту конфигурацию можно с помощью программы `sysinstall`

1. Перейдите в режим суперпользователя и введите **`/stand/sysinstall`** чтобы запустить программу
2. Выберите `Configure`, чтобы перейти в меню конфигурирования FreeBSD
3. Выберите `Networking`, чтобы перейти в меню конфигурирования сети
4. Выберите `Ntpdate`, чтобы активизировать утилиту `ntpdate` Программа отобразит меню `NTPDATE Server Selection`, в котором будет перечислено множество NTP-серверов, в основном относящихся к уровню 2
5. Выберите сервер из списка (впоследствии можно будет задать сервер, отсутствующий в списке), после чего снова появится меню конфигурирования сети
6. Последовательно выберите `Cancel`, `Cancel` и `Exit Install`, чтобы выйти из программы

В файл `/etc/rc.conf` будут добавлены следующие строки

```
ntpdate_flags="clock-2 cs cmu.edu"
ntpdate_enable="YES"
```

Их можно ввести и вручную не прибегая к помощи программы `sysinstall` Можно также задать сервер, не указанный в списке. Скорее всего именно так придется поступить, чтобы ввести адрес главного локального сервера

В ходе очередной перезагрузки FreeBSD свяжется с сервером и вызовет утилиту `ntpdate` для синхронизации времени с сервером Если демон `ntpd` не запущен, то же

самое можно делать и вручную с помощью команды `ntpdate` *имя_сервера*. Создайте для этой команды задание демона `cron` чтобы время настраивалось, скажем раз в день, если расхождение на несколько секунд в течение дня допустимо

Конфигурирование NTP-сервера

Чтобы непрерывно контролировать время, нужно запустить демон `ntpd`. Для этого необходимо выполнить два действия отредактировать файл `/etc/rc.conf`, чтобы FreeBSD запускала демон на этапе начальной загрузки, и задать в файле `/etc/ntp.conf` нужные параметры демона, включая адрес используемого сервера синхронизации.

В файл `/etc/rc conf` нужно добавить следующую строку

```
xntpd_enable="YES"
```

После перезагрузки демон активизируется, свяжется с родительскими серверами и начнет корректировать ход системных часов Демон можно запустить и вручную введя `ntpd` в режиме суперпользователя

Конфигурационный файл демона называется `/etc/ntp conf` Наиболее важный параметр находится в начале файла·

```
server clock-2 cs.cmu.edu
```

Эта строка сообщает системе адрес родительского NTP-сервера. Можно включить несколько таких строк, по одной для каждого сервера, тогда демон свяжется с каждым из них и выберет наиболее точный источник В стандартной конфигурации содержится ссылка на узел 127.127 1 0 Это IP-адрес, закрепляемый демоном за локальным таймером на тот случай, если все серверы окажутся недоступны (в другой строке, начинающейся с ключевого слова `fudge`, локальному таймеру назначается уровень 10 чтобы он никогда не участвовал в процедуре опроса настоящих серверов).

Если протокол NTP должен использоваться во всей сети, только один из компьютеров нужно сконфигурировать на синхронизацию с внешними NTP-серверами. Внутренние компьютеры будут иметь такую же конфигурацию, но ссылаться на главный локальный NTP-сервер Последний должен относиться к уровню, номер которого на единицу больше, чем у родительского сервера. Номер уровня остальных систем еще на единицу выше. Например, если главный компьютер синхронизируется с сервером уровня 2, то он принадлежит к уровню 3, а остальные системы являются серверами уровня 4 (хотя у них и нет клиентов)

К сожалению, не всегда удается определить, работает ли демон `ntpd` так, как ожидается Часто сервер избегает корректировать время скачкообразно Вместо этого он меняет скорость хода часов, пока время не станет точным Кроме того, сервер может полностью прекратить работу, если окажется, что время смещено больше чем на час Чтобы узнать, работает ли сервер, нужно ввести **`ps ax | grep ntpd`**, а для проверки корректной работы сервера необходимо воспользоваться утилитой мониторинга `ntpq` Она поддерживает много директив (вводятся в строке приглашения `ntpq>`), наиболее важная из которых — `peers` Ее применение продемонстрировано на рис 22.1. Сервер, имя которого помечено звездочкой в левой колонке, выбран демоном `ntpd` для синхронизации часов Серверы, помеченные знаком '+', тоже опрашивались и были признаны корректными, однако по какому-то критерию не выбраны (значения критериев приведены в трех последних колонках)

На заметку

Сразу после запуска демону `ntpd` требуется несколько минут чтобы решить, с каким из внешних серверов следует синхронизироваться. В этот промежуток времени ни один из серверов не помечается звездочкой. Остальные значения, сообщаемые утилитой `ntpq`, тоже будут меняться и клиенты не будут иметь возможности подключиться к серверу

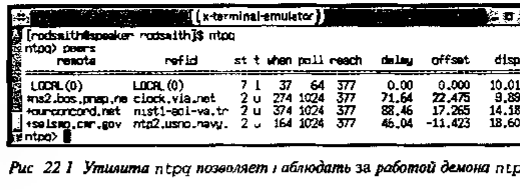


Рис. 22.1. Утилита `ntpq` *позволяет наблюдать за работой демона* `ntpd`

Когда во FreeBSD работает демон `nptd` NTP клиенты других операционных систем могут взаимодействовать с ним. Такие клиенты есть во многих системах, включая Windows XP и Mac OS X. Информацию о различных клиентах и серверах можно найти по адресу `http://www.ntp.org/software`

Часто сервер синхронизации времени используется только для нужд локальной сети. Это означает, что на брандмауэре нужно задать правила фильтрации пакетов, блокирующие доступ к порту 123 серверного компьютера со стороны неавторизованных клиентов.

Серверы шрифтов

Последний класс серверов, рассматриваемых в этой главе — серверы шрифтов. Это программы, предоставляющие информацию о шрифтах компьютерам. Шрифт инсталлируется один раз, на сервере, после чего становится доступен всем X-серверам сети, при условии, что они сконфигурированы на подключение к серверу шрифтов. В крупной сети с большим количеством рабочих станций это существенно упрощает конфигурирование X среды на каждой из них, особенно если набор доступных шрифтов часто меняется.

Правовой статус шрифтов зависит от страны. В США шрифты нельзя защищать авторским правом, хотя форматы файлов TrueType и Adobe Type 1 технически подпадают под категорию компьютерных программ и, следовательно, могут быть защищены. В любом случае узнайте у поставщика шрифта, разрешается ли делать его доступным через сервер шрифтов.

Обработка шрифтоа в X-среде

Чтобы разобраться с серверами шрифтов, необходимо понять, как вообще шрифты обрабатываются в X Window System. X-среда появилась, когда процессорные мощности были слабы, а объем памяти, по современным стандартам, был ограничен. В те времена часто использовались *растровые шрифты*, символы которых описывались наборами пикселей. Растровый шрифт может иметь только один размер, и если масштабируется, то искусственным образом. Такие шрифты приходилось настраивать вручную, чтобы они корректно отображались при разном разрешении экрана. Некоторые X шрифты до сих пор поставляются в растровом формате, поскольку они лучше выглядят при очень маленьком разрешении, чем шрифты более современных форматов, обрабатываемые компьютером. Это объясняется тем, что вручную шрифт легче настроить в соответствии с эстетическими особенностями восприятия.

В последнее десятилетие широкую популярность приобрели *масштабируемые* (или *контурные*) шрифты. Они описываются математическим способом, в виде набора линий и кривых, определяющих контуры каждого символа. Компьютер заполняет внутреннее пространство символа пикселями нужного цвета и самостоятельно определяет какие пиксели следует отображать при выводе шрифта заданного размера. Наиболее распространенными форматами масштабируемых шрифтов являются PostScript Type 1 компании Adobe (его еще называют Adobe Type Manager, или ATM) и TrueType компании Apple. Оба формата поддерживаются программой XFree86 версии 4.*x*. В XFree86 3.3.*x* и более ранних версий шрифты TrueType не поддерживались.

Сервер шрифтов избавляет X-сервер от необходимости выполнять визуализацию шрифтов (т.е. преобразовывать символы из контурного формата в растровый перед отображением на экране). Сервер шрифтов создает растровый образ шрифта и передает его X-серверу. Оба сервера могут работать на одном компьютере, но в этой главе мы рассмотрим конфигурацию автономного сервера.

Если используется старая программа XFree86 3.3.*x* и нет возможности перейти на версию 4.*x*, инсталлируйте локальный сервер шрифтов, способный обрабатывать шрифты TrueType. Подойдет сервер пакета XFree86 версии 4.*x*, сервер X-TT (`http://www.ic.com/~kazushi/xtt/`) или `xfsft` (`http://www.dcs.ed.ac.uk/home/jec/programs/xfsft/`).

Обычно X-сервер ищет файлы шрифтов в каталогах, заданных в файле `XF86Config` (см. главу 13). Если нужно использовать сервер шрифтов, укажите его адрес в одной из переменных `FontPath`, как будет описано на с. 446.

Даже при наличии сервера шрифтов нежелательно удалять из списка каталоги стандартных шрифтов. Если сервер вдруг станет недоступен, компьютер сможет использовать стандартные шрифты. По крайней мере этого достаточно, чтобы выгрузить X среду.

Конфигурирование сервера шрифтов

Стандартный сервер шрифтов FreeBSD инсталлируется как часть пакета `XFree86-FontServer`, установленного в системе по умолчанию. Серверный файл называется `/usr/X11R6/bin/xfs`.

Создание каталогов шрифтов

Сервер шрифтов должен иметь доступ к файлам шрифтов, хранящимся в специальных каталогах. К примеру, можно создать каталог `/usr/local/fonts`, разделив его на подкаталоги по типам шрифтов, как это сделано в стандартном каталоге `/usr/X11R6/lib/X11/fonts` (подкаталоги `Speedo`, `Type1` и т.д.). В то же время можно воспользоваться существующими каталогами шрифтов, сделав их доступными через сервер шрифтов.

Шрифты инсталлируются и конфигурируются точно так же, как и при непосредственном их использовании программой XFree86 (см. главу 13). Не забудьте создать файл `fonts.dir` с описанием шрифтов. Если стандартные шрифты XFree86 делаются доступными через сервер шрифтов, то никакие изменения каталогов не нужны.

Изменение конфигурационного файла

Конфигурационный файл сервера шрифтов называется `config`. Во FreeBSD 5.0 он находится в каталоге `/usr/X11R6/lib/X11/fs`. В этом файле задаются различные па-

раметры менять которые не следует, так как они определяют использование памяти стандартные размеры шрифтов и т.д. Параметр, на который следует обратить внимание, называется `catalogue` В нем перечислены каталоги шрифтов, контролируемые сервером Это эквивалент группы переменных `FontPath` в файле `X86Config`, но здесь все значения объединены в одну строку, элементы которой разделяются запятыми Короткий пример выглядит так:

```
catalogue = /usr/X11R6/lib/X11/fonts/TTF,/usr/local/fonts/type
```

Здесь подключаются шрифты, находящиеся в двух каталогах на сервере. Изначально в параметре `catalogue` указано несколько дополнительных подкаталогов стандартного каталога `/usr/X11R6/lib/X11/fonts/`, что позволяет совместно использовать шрифты, принятые во FreeBSD по умолчанию. Это удобно, если доступ к серверу шрифтов осуществляется из X-сервера, работающего не в среде FreeBSD, а, например, в Windows, где набор шрифтов меньше. Поскольку переменная `catalogue` одна, соответствующая строка файла может стать довольно длинной Убедитесь в том, что текстовый редактор не разбивает длинные строки автоматически.

Следует также учитывать параметр `port`, в котором задается TCP-порт, прослушиваемый сервером. Значение по умолчанию — 7100, но бывают случаи, когда нужно использовать другой порт Например, некоторые программы содержат специализированные серверы шрифтов, захватывающие порт 7100 Если это так включите в файл `config` строку следующего вида

```
port = 7101
```

Запуск сервера шрифтов

После задания конфигурационных настроек нужно запустить сервер шрифтов Для однократного вызова сервера достаточно ввести его имя. Необязательный аргумент `daemon` переводит сервер в режим демона

```
# /usr/X11R6/bin/xfs -daemon
```

Теперь можно сконфигурировать локальную программу XFree86 на использование сервера шрифтов, как описывается далее Все остальные компьютеры тоже смогут получить доступ к серверу, если только в сети не заданы специальные правила фильтрации

Сервер шрифтов не должен быть доступен компьютерам в Internet За претите такой доступ с помощью брандмауэра

Чтобы сервер `xfs` был доступен постоянно, он должен запускаться на этапе начальной загрузки системы Стандартные сценарии запуска FreeBSD не предусматривают такую возможность, поэтому нужно самостоятельно создать сценарий `/etc/rc.local` или `/usr/local/etc/rc d/xfs.sh` (последний предпочтительнее в современных версиях FreeBSD). В простейшем случае в сценарий достаточно включить показанную выше команду запуска Кроме того, у сценария должен быть установлен бит выполнения (подробнее об этом рассказывалось в главе 6)

Испольвование сервера шрифтов

В главе 13 говорилось о том, как конфигурировать шрифты X сервера и там же было показано, как в файле `/etc/X11/XF86Config` задать ссылку на сервер шрифтов Напомним, что в раздел `Files` нужно включить строку следующего вида.

```
FontPath   "tcp/сервер_шрифтов:порт"
```

Здесь *сервер_шрифтов* – это доменное имя или IP-адрес серверного компьютера, а *порт* — это номер порта, на котором работает сервер. Лучше всего вставить эту строку

после определений локальных каталогов, так как доступ к последним осуществляется быстрее и к тому же программа XFree86 просматривает каталоги в том порядке, в котором они перечислены. Чтобы X-сервер использовал сервер шрифтов, необходимо перезапустить X-среду. Если же нужно подключить сервер шрифтов немедленно, введите следующие команды.

```
$ xset fp+ tcp/сервер_шрифтов порт
$ xset fp rehash
```

Эти команды заставят X-сервер добавить указанный сервер к списку шрифтов и повторно загрузить весь список.

Если переконфигурировать шрифты не удалось, возможно, сервер шрифтов не отвечает. Это означает, что, прежде чем пытаться изменять конфигурацию шрифтов X-сервера, нужно сохранить открытые файлы или даже закрыть выполняющиеся программы.

Сервер шрифтов FreeBSD способен пересылать шрифты X серверам, отличным от XFree86. Если используется X-сервер Windows или Mac OS, к нему, как правило, можно подключить сервер шрифтов. Обычно такие программы предоставляют диалоговое окно, в котором можно задать каталоги и серверы шрифтов.

Резюме

Работу сетей обеспечивает множество серверов. Пользователям редко приходится с ними сталкиваться, но эти серверы являются важными компонентами сетевой инфраструктуры. DHCP-серверы предоставляют IP-адреса и прочую конфигурационную информацию клиентским компьютерам, осуществляя динамическую настройку сетевой подсистемы. Серверы DNS преобразуют имена компьютеров в IP-адреса и наоборот. Серверы синхронизации времени настраивают часы всех компьютеров в сети. Серверы шрифтов упрощают конфигурирование шрифтов в сети с большим количеством X-серверов. Стандартная конфигурация этих серверов подходит для небольших сетей, но некоторые параметры приходится менять с учетом особенностей сети.

Полный справочник по

Часть V

Пользовательское программное обеспечение

Полный справочник по

Глава 23

Настольные графические оболочки

Этой главой начинается часть V, посвященная пользовательским приложениям Те же сетевые клиенты (описываются в главе 24) — это пользовательские программы, являющиеся одними из важнейших программных компонентов с точки зрения пользователей

В данной главе рассматриваются настольные оболочки, в которых выполняются пользовательские программы. В современных рабочих станциях настольные оболочки являются, как правило, графическими; во FreeBSD такого рода оболочка представляет собой тесно интегрированный набор программ Мы познакомимся с основными доступными графическими оболочками, уделив основное внимание оболочке GNOME Но сначала нужно разобраться с тем, что собой представляет настольная оболочка и зачем она нужна

Роль графических оболочек

Графическая среда FreeBSD — X Window System (сокращенно *X среда*) - в чем то сложнее и в чем-то проще графических оболочек других операционных систем, таких как Windows и Mac OS Сложность X-среды в том, что она изначально ориентирована на сетевое применение, благодаря чему X программы можно запускать с других компьютеров Простота же среды объясняется тем, что в ней практически отсутствуют встроенные графические утилиты и вспомогательные программы. X-среда предоставляет в распоряжение программистов средства отображения окон, рисования линий, окружностей и других графических примитивов в окнах, приема данных от пользователей в виде нажатий клавиш и щелчков мыши. Но сама среда не содержит высокоуровневые графические конструкции, такие как меню, кнопки и т д Все они относятся к X-наборам элементов управления — программным библиотекам, позволяющим реализовывать средства пользовательского интерфейса поверх X-среды (см главу 13). В X Window System отсутствуют и вспомогательные утилиты, имеющиеся в конкурирующих системах· калькуляторы, простейшие редакторы, средства просмотра файлов и т д Эти программы входят в состав графических оболочек

В целом настольная графическая оболочка состоит из следующих компонентов

- **Диспетчер окон** - программа, позволяющая пользователям работать с окнами. Диспетчеры окон формируют *строку заголовка*, отображаемую в верхней части окна и используемую для перетаскивания окна по экрану. Кроме того, диспетчеры окон отображают всплывающие меню из которых можно запускать программы.
- **Диспетчер файлов** — создает окна, отображающие списки файлов в конкретных каталогах. При двойном щелчке на значке файла в таком окне запускается программа, связанная с соответствующим типом файлов Диспетчер файлов позволяет копировать и перемещать файлы создавать новые файлы и каталоги и т.д
- **Средства управления экраном** Настольные оболочки обычно включают одну или несколько утилит для управления экраном — задания фонового цвета или изображения, переключения между *виртуальными рабочими столами* (это параллельные виртуальные экраны, позволяющие запускать множество программ, не загромождая экран большим количеством окон) и т д Некоторые диспетчеры окон тоже содержат аналогичные средства, однако включение их в настольную оболочку означает, что она будет работать независимо от диспетчера окон

- **Средства настройки пользовательского интерфейса** В настольную оболочку входят конфигурационные утилиты, позволяющие задавать параметры пользовательского интерфейса, такие как скорость перемещения указателя мыши, скорость повтора при нажатии клавиш и стандартные шрифты используемые интегрированными программами
- **Служебные утилиты** В состав настольных оболочек входят небольшие программы, такие как калькуляторы, терминальные программы утилиты набора номера текстовые редакторы, простейшие игры и т.д.
- **Основные интегрированные приложения** Настольные оболочки дополняются такими приложениями как Web-броузеры, почтовые клиенты текстовые процессоры, редакторы электронных таблиц и т.д.

Настольные оболочки не обладают монополией на все эти программы. Как будет показано на с. 457, можно смешивать компоненты различных оболочек и добавлять к ним программы из внешних источников, формируя настольную оболочку, которая лучше приспособлена к потребностям пользователя, чем готовая система. Но у стандартных оболочек свои преимущества Одно из них — простота. Пользователю-новичку проще сконфигурировать систему на использование, допустим, оболочки GNOME чем выбирать нужные среди тысячи доступных компонентов, создавая из них рабочую среду Другим преимуществом интегрированных оболочек является то что их компоненты проектировались в расчете на тесное взаимодействие. Каждая графическая оболочка реализована на основе единого набора элементов управления поэтому все компоненты имеют согласованный вид Кроме того, отдельные компоненты часто используют установки по умолчанию, применимые ко всем программам например стандартный шрифт строки меню или стандартную цветовую схему В некоторых случаях задействуются единые форматы файлов в частности для интегрированных адресных книг

Часто графические оболочки инсталлируются в виде набора пакетов Сюда входят пакеты для библиотек базовых функций и вспомогательных программ, пакеты для основных групп программ (сетевые утилиты, игры и т д) и пакеты для интегрированных приложений (текстовые процессоры, Web броузеры и др.). О наиболее важных прикладных пакетах будет рассказываться в последующих главах В частности, в главе 24 описываются сетевые утилиты, такие как почтовые клиенты и Web-броузеры, в главе 25 — офисные средства, в том числе текстовые процессоры и электронные таблицы в главе 26 — средства компьютерной графики, в главе 27 — мультимедийные приложения и игры.

Как правило, программы, написанные для одной оболочки или входящие в ее состав, можно запускать в другой оболочке Например, GIMP — популярный графический редактор, являющийся официальным компонентом оболочки GNOME, но он работает и в KDE, и в XFce, и в пользовательских оболочках То же самое справедливо и для более специализированных программ, таких как Web-броузер Konqueror оболочки KDE или программа Terminal оболочки GNOME. Иногда программа теряет часть функциональных возможностей при работе вне своей родительской оболочки Например, могут не поддерживаться операции перетаскивания между программами из разных оболочек. В редких случаях программа отказывается работать или становится бесполезной, но это обычно относится лишь к утилитам конфигурирования самих оболочек

На заметку

Иногда о программе говорят "GNOME-программа" или "KDE программа", хотя на самом деле программа просто использует тот же набор элементов управления, что и соответствующая графическая оболочка Такая программа может содержать средства позволяющие ей интегрироваться с оболочкой но в принципе она способна работать и независимо от нее

Обзор доступных графических оболочек

Наиболее популярными графическими оболочками для FreeBSD являются GNOME (GNU Network Object Model Environment) и KDE (K Desktop Environment). Обе они включают полный набор настольных утилит и приложений. В настоящее время по умолчанию инсталлируется GNOME, но при желании можно установить KDE. Обе оболочки занимают десятки мегабайт оперативной памяти, поэтому они могут оказаться слишком громоздкими для систем с ограниченным объемом ОЗУ. Третья оболочка – XFce – занимает меньше памяти, зато проигрывает по функциональным возможностям. Она смоделирована на основе коммерческой оболочки CDE (Common Desktop Environment), популярной в некоторых коммерческих UNIX системах. Можно создавать и собственные настольные оболочки, объединяя необходимые компоненты из различных источников. Такой подход позволяет получить менее ресурсоемкую оболочку, хотя для этого придется приложить немало усилий.

На заметку Если в системе работает большое число пользователей, инсталлируйте несколько графических оболочек и предоставьте пользователям возможность выбора. О том, как это сделать, рассказывалось на с. 168.

KDE

KDE (`www.kde.org`) была первой интегрированной графической оболочкой для FreeBSD. Она основана на оконном интерфейсе Qt – одном из ведущих наборов элементов управления для X Window System. KDE состоит из следующих компонентов:

- **Диспетчер окон**. В KDE используется диспетчер KWin, который столь тесно интегрирован, что подключить другой диспетчер вряд ли удастся.
- **Диспетчер файлов и Web-броузер**. Программа Konqueror используется в качестве диспетчера файлов, а заодно и Web-броузера. Эту программу легко запустить из меню KDE.
- **Настольные средства**. В KDE используется стандартный набор настольных средств, таких как экранные меню для запуска программ, экранные часы, конфигурационные программы, почтовый клиент и т. д.
- **Офисный пакет**. KDE включает в себя пакет офисных средств KOffice, состоящий из текстового процессора, программы обработки электронных таблиц и других компонентов. Подробнее этот пакет описывается в главе 25.

Помимо пользовательских средств KDE содержит средства, которые упрощают разработку приложений, легко интегрирующихся с оболочкой. Применение этих средств позволяет, к примеру, реализовывать операции перетаскивания объектов между KDE-приложениями. Еще один инструмент разработчика KDE — это средства поддержки *сглаженных шрифтов*. Методика сглаживания заключается в использовании полутоновых пикселей при отображении кривых или диагональных линий символов, что создает иллюзию большего разрешения экрана, чем есть на самом деле. Режим сглаживания шрифтов включается в разделе Look & Feel⇨Fonts программы Control Center.

Версия KDE 3.0 появилась в 2002 году и напоминает более ранние версии 2.2.*x*. На рис. 23.1 представлен экран сеанса KDE 3.0. Здесь демонстрируются ключевые особенности KDE, многие из которых характерны и для других графических оболочек. На рис. 23.1 присутствуют окна программы Control Center (используется для конфигурирования KDE), броузера Konqueror и терминальной программы Konsole. В левом верхнем углу видны значки рабочего стола KDE. Они позволяют получить быстрый доступ к программам, каталогам компьютера или Web-узлам.

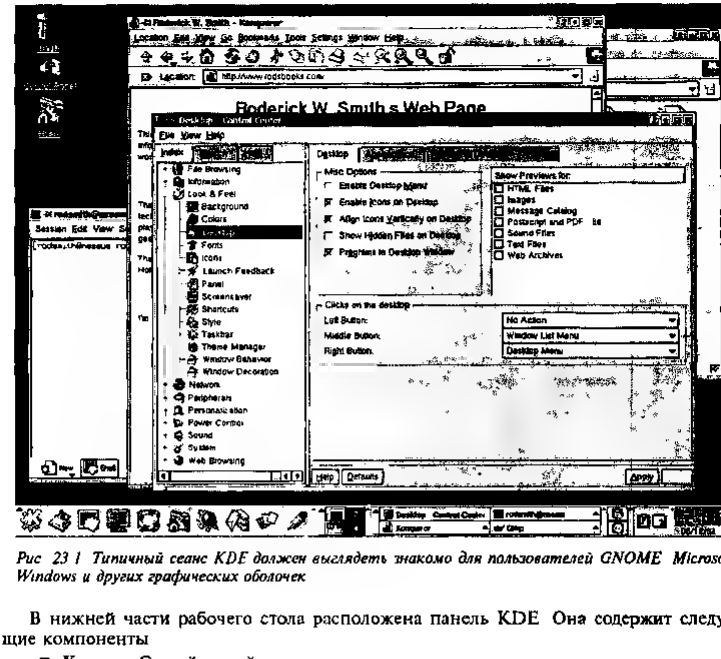


Рис 23 1 Типичный сеанс KDE должен выглядеть знакомо для пользователей GNOME Microsoft Windows и других графических оболочек

В нижней части рабочего стола расположена панель KDE Она содержит следую щие компоненты

- **К-меню**. Самый левый значок панели позволяет получить доступ ко множеству программ FreeBSD, подобно стартовому меню Windows.
- **Значки быстрого запуска программ**. Следующие несколько значков (в данном случае — девять) служат для вызова важных или часто используемых программ таких как Konsole, Control Center и Konqueror.
- **Меню переключения экранов**. В KDE есть меню, позволяющее переключаться между виртуальными рабочими столами На рис. 23.1 в системе сконфигурировано 4 рабочих стола. Чтобы выбрать нужный, щелкните на его миниатюрном обозначении в панели KDE
- **Значки открытых окон**. Чтобы перейти к программе у которой открыто окно в KDE, щелкните не ее значке в панели KDE.
- **Команды выхода и блокировки экрана**. После значков открытых окон располо жены небольшие значки, напоминающие выключатель и замок. Они предна значены для завершения сеанса и блокировки экрана.
- **Мини утилиты**. В KDE входит ряд небольших утилит, позволяющих к приме ру просмотреть содержимое буфера обмена или календарь. Некоторые из них доступны через панель KDE
- **Часы**. В правой части панели отображаются часы. Щелкните на них левой кнопкой мыши, чтобы отобразить календарь и правой кнопкой, чтобы на строить работу часов.

- **Меню настройки панели** По умолчанию панель всегда отображается. Если щелкнуть на треугольном значке в правом углу панели, она будет скрыта, что позволит использовать больше экранного пространства для окон программ. С помощью программы Control Center можно так сконфигурировать панель, что бы она появлялась только при перемещении указателя мыши в нижнюю часть экрана

Многие характеристики KDE допускают конфигурирование Можно добавлять и удалять значки на рабочем столе, добавлять и удалять элементы панели задач, менять внешний вид и детали работы диспетчера KWin и многое другое В основном изменения осуществляются через программу Control Center, но есть и другие способы. Например в меню K⇨Configure Panel содержатся команды настройки панели KDE

GNOME

GNOME — очень популярная графическая оболочка во FreeBSD. Именно она будет взята за образец при изучении особенностей графических оболочек в следующем разделе. Своим появлением оболочка GNOME обязана тому факту, что первая из появившихся оболочек — KDE — основана на оконном интерфейсе Qt. На тот момент условия лицензирования Qt не позволяли считать этот интерфейс полностью открытым, поэтому многие сторонники свободно распространяемого программного обеспечения избегали использовать Qt-программы (с тех пор условия лицензирования изменились, и теперь Qt имеет открытый исходный код) Так возник проект GNOME, основанный на оконном интерфейсе GTK+ (GIMP Tool Kit), который первоначально был разработан для графического редактора GIMP.

Набор компонентов GNOME во многом аналогичен тому, что есть в KDE

- **Диспетчер окон** В отличие от KDE, в GNOME доступно большое число диспетчеров окон В стандартной конфигурации используется программа Sawfish На практике лучше всего работают диспетчеры, в которых предусмотрена встроенная поддержка GNOME такие как Sawfish, AfterStep Window Maker Enlightenment и IceWM
- **Диспетчер файлов** Стандартный диспетчер файлов называется Nautilus. В отличие от программы Konqueror в KDE, Nautilus не дублирует функции Web броузера
- **Web-броузер** Встроенным броузером GNOME является программа Galeon. Она ведет начало от броузера Mozilla — открытого "собрата" Netscape Navigator Впрочем, в GNOME можно запустить любой другой броузер
- **Настольные средства** В GNOME используется стандартный набор настольных средств таких как экранные меню для запуска программ экранные часы, конфигурационные программы, почтовый клиент и т д
- **Офисный пакет** В GNOME несколько несвязанных ранее офисных программ были объединены в пакет GNOME Office. В настоящее время ведутся работы по улучшению их взаимодействия Подробнее пакет GNOME Office описывается в главе 25

В середине 2002 года была выпущена версия GNOME 2 0 В ней используются новые внутренние структуры данных, поэтому она плохо совместима с предыдущими версиями. Не рекомендуется смешивать компоненты версий 2 0 и выше с программами из более ранних версий GNOME

На рис 23 2 приведен экран сеанса GNOME Сравнивая с рис. 23 1, можно заметить что оболочка GNOME весьма схожа с KDE, по крайней мере в первом приближении Обе оболочки содержат панели задач из которых можно вызывать программы,

переключаться между программами и между виртуальными рабочими столами. Как и в KDE, в GNOME есть программа Control Center (окно переднего плана на рисунке) предназначенная для конфигурирования оболочки.

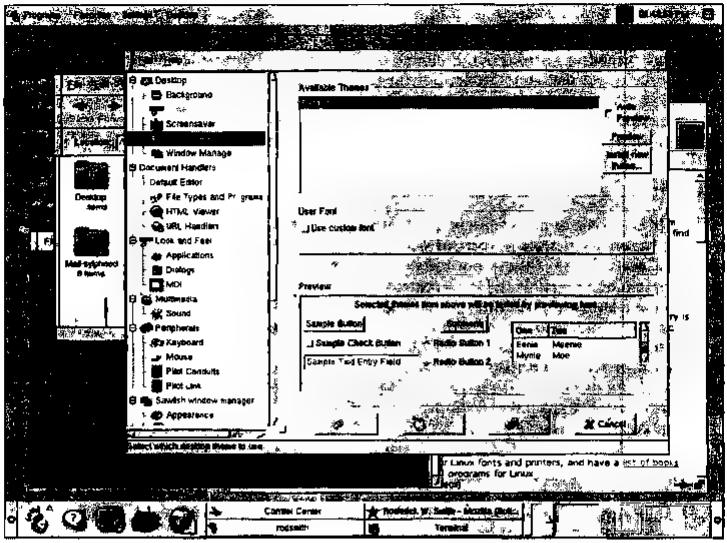


Рис 23 2. Рабочий стол GNOME содержит элементы схожие с теми, что используются в KDE

Многие аспекты отличают GNOME от KDE Диспетчер окон GNOME удачнее сконфигурирован, чем в KDE (рис 23 2 сделан в диспетчере IceWM, тема Helix) Кнопки, меню и другие элементы интерфейса выглядят по-разному, ведь оболочка GNOME основана на оконном интерфейсе GTK+, а KDE основана на Qt Отдельные компоненты, такие как калькуляторы и редакторы, имеют несколько иной набор функциональных свойств В стандартной конфигурации GNOME панель задач ото бражается в нижней части экрана а стартовое меню — в верхней, причем часть их функций дублируется

Сложно отдать предпочтение одной из оболочек Это во многом дело вкуса Пора ботайте неделю-другую с каждой из них и решите для себя какая оболочка для вас удобнее

XFce

XFce – это небольшая графическая оболочка, уступающая KDE и GNOME как по размеру, так и по популярности В ее основе лежит оконный интерфейс GTK+ (рис. 23.3). Если разработчики GNOME и KDE ориентировались на графические обо лочки таких систем, как Windows, OS/2 и Mac OS Classic, то разработчики XFce мно гое заимствовали из коммерческой оболочки CDE используемой в UNIX Перечис лим основные компоненты XFce

- **Диспетчер окон** В XFce используется собственный диспетчер окон XFwm. Он не так хорошо конфигурируется, как другие диспетчеры, зато вполне функционален.
- **Диспетчер файлов** В XFce есть диспетчер файлов XFTree. Подобно многим другим компонентам оболочки, он уступает своим аналогам в KDE или GNOME, но обладает всеми необходимыми функциональными возможностями.
- **Настольные средства**. В XFce есть несколько настольных средств. Некоторые из них, в частности панель задач, специфичны для XFce. Другие, как, например калькулятор — это обычные X программы, сконфигурированные для работы в XFce.

В XFce нет офисных средств, но можно инсталлировать пакеты GNOME Office, KOffice и другие. Нет здесь и специализированного Web-броузера. Вместо этого используется броузер Mozilla. В состав оболочки входит гораздо меньше конфигурационных и служебных утилит, чем в KDE или GNOME, и многие встроенные программы уступают по своим функциональным возможностями эквивалентным утилитам других оболочек. Большим преимуществом XFce, особенно для компьютера с ограниченным объемом оперативной памяти, является небольшой размер оболочки.

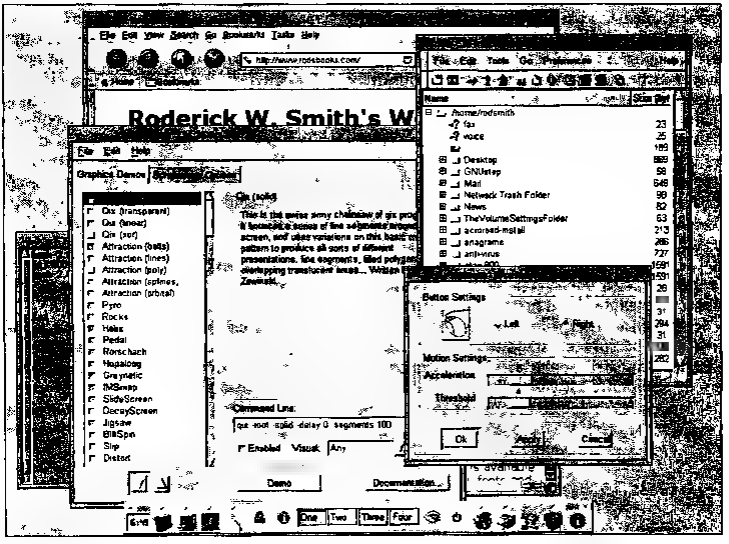


Рис. 23.3. В XFce меньше средств конфигурирования и управления рабочим столом, чем в KDE или GNOME, зато это менее ресурсоемкая среда

Запуск собственной графической оболочки

Одним из достоинств X Window System является ее модульность, благодаря которой пользователи могут настраивать рабочие столы так, как это невозможно сделать в других графических средах. Даже если вас не устраивает KDE, GNOME, XFce или

другая готовая графическая оболочка, ничто не мешает создать собственную оболочку Конечно, это не самая простая задача для новичка UNIX, но и не настолько сложная как может показаться на первый взгляд Никаких программ писать не нужно. Доста точно лишь подобрать необходимые компоненты и объединить их

Начните с диспетчера окон. Многие из них включены в состав FreeBSD в виде па кетов или переносимых приложений (раздел `x11-wm`) Для простой оболочки подойдут диспетчеры IceWM, Window Maker, `fvwm` и его варианты, Sawfish и Black Box. Тем кому нужен максимально конфигурируемый и гибкий диспетчер окон, порекоменду ем Enlightenment

Собственно говоря, диспетчер окон — это все что необходимо для формирования простейшей графической оболочки Вспомогательные компоненты можно вызывать из того же файла, который используется для запуска самого диспетчера как описыва лось в главе 6. Диспетчер файлов можно позаимствовать из KDE, GNOME или XFce Существуют и независимые диспетчеры файлов, такие как Desktop File Manager, или DFM (`http://www.kaisersite.de/dfm/`), Gentoo (`http://www.obsession se/gentoo/`) и TkDesk (`http://tkdesk.sourceforge.net`) Образцы программ этого класса можно найти в разделе `x11-fm` коллекции пакетов или переносимых приложений.

Далее необходимо подобрать вспомогательные программы составляющие основу рабочей среды: калькуляторы, почтовые клиенты редакторы и т д Многие из них входят в состав FreeBSD, поэтому их несложно найти и установить с помощью про граммы `sysinstall`. Чтобы сделать эти программы доступными в графической среде следует отредактировать конфигурационный файл диспетчера окон или диспетчера файлов. Как правило, эти файлы называются так же, как и сам диспетчер, или хранятся в одноименном подкаталоге начального каталога пользователя. Например, глобальные установки диспетчера окон IceWM хранятся в файле `~/.icewm/preferences`, а описа ние его меню находится в файле `~/.icewm/menu` Последний можно редактировать добавляя команды запуска нужных программ в стартовое меню. Диспетчер файлов DFM использует файл `~/.dfmext` для привязки определенных типов файлов к про граммам обработчикам, чтобы программу можно было вызвать двойным щелчком на значке файла. В каталоге `~/.dfmdesk` содержатся файлы, которым сопоставлены значки на рабочем столе. У DFM есть собственные контекстные меню, с помощью которых можно добавлять значки программ на рабочий стол; при каждой такой опе рации в каталоге `~/.dfmdesk` создается ссылка или файл.

Некоторые инструментальные средства, входящие в состав интегрированных гра фических оболочек, например утилиты конфигурирования мыши реализуются в пользовательских оболочках с помощью гораздо более простых программ, вызывае мых из сценария запуска оболочки В частности, программа `xset` позволяет регулиро вать скорость перемещения указателя мыши и скорость повтора а программа `xsetroot` может задавать цвет фона Соответствующий фрагмент файла `xinitrc` или `.xsession` выглядит так:

```
xset m 4
xset r 250 15
xsetroot -solid SteelBlue
```

В первой строке скорость перемещения указателя устанавливается равной 4 (точный эффект этой команды зависит от мыши и монитора). Во второй строке вклю чается режим автоматического повтора символа при удержании клавиши, причем на чальная задержка равна 250 миллисекунд, а скорость повтора составляет 15 символов в секунду В третьей строке цвет фона делается синим Дополнительную информацию о программах `xset` и `xsetroot` можно найти на их `man`-страницах

Формируя собственную графическую оболочку, можно гораздо точнее смодели ровать желаемую рабочую среду, чем в случае готовой оболочки. KDE и GNOME имеют широкие функциональные возможности, но они весьма требовательны к ре

сурсам памяти и могут не вполне соответствовать потребностям пользователя. В целом пользовательские оболочки имеет смысл создавать в системах с ограниченными ресурсами

Работа в среде GNOME

GNOME — это популярная настольная среда во FreeBSD По составу и функциональным возможностям она очень напоминает KDE, хотя есть некоторые различия. В этом разделе будут рассмотрены особенности работы в среде GNOME, включая запуск оболочки и конфигурирование рабочего стола Отдельные аспекты функционирования оболочки рассматривались в разделе "Краткий обзор графической среды" главы 3

Запуск GNOME

GNOME представляет собой приложение FreeBSD, точнее, набор программ. Это значит, что теоретически для запуска оболочки достаточно ввести имя ее основной программы (`gnome-session`) в терминальном окне Впрочем, на практике GNOME, как и любая другая графическая оболочка, вызывается из сценария запуска X-среды Посредством такого сценария активизируются ключевые программы, которые всегда должны работать в X-среде. Наиболее распространенные сценарии запуска — `~/.xinitrc` (применяется, когда пользователь регистрируется в текстовом режиме и запускает X-среду командой **`startx`**), `~/.xsession` (используется графической регистрационной программой XDM) и `~/.vnc/xstartup` (используется сервером дистанционной регистрации VNC, который описывался в главе 21) Все они работают одинаково, в них содержатся команды, выполняемые при запуске X-среды или регистрации в ней Если программа запускается в фоновом режиме, в конце командной строки указывается оператор `&` Последняя команда обычно служит для вызова диспетчера окон или графической оболочки Амперсанд в конце не ставится, поэтому после выхода пользователя из оболочки сценарий продолжит работу и X-сеанс завершится Минимальный сценарий регистрации содержит одну-единственную строку — `gnome session`, при условии, что нужно запустить оболочку GNOME

На заметку Иногда после команды вызова диспетчера окон или графической оболочки встречаются дополнительные строки Эти команды осуществляют операции очистки, такие как удаление временных файлов, или проигрывают звуковой фрагмент, ассоциируемый с выходом из системы

При первом запуске оболочки на экране появляется диалоговое окно GNOME Hint в котором даются советы по работе в среде GNOME Если вы не хотите, чтобы это окно появлялось в будущем, сбросьте флажок Display This Dialog Next Time и щелкните на кнопке Close. Экран собственно графической оболочки GNOME практически пуст Он выглядит примерно так, как на рис. 23 2, но без открытых окон и, возможно без значков на рабочем столе Для вызова программы нужно щелкнуть на ее значке в панели задач (нижняя часть экрана) или выбрать ее название в меню Programs (верхняя часть экрана)

Чтобы завершить работу в графической среде, нужно выбрать команду Log Out в стартовом меню (GNOME Foot — значок с изображением ступни) или в меню Desktop Результат зависит от способа регистрации в системе: произойдет возврат к экрану регистрации в текстовом режиме либо к экрану регистрации программы XDM или же окно регистрации сервера VNC исчезнет Далее можно снова войти в систему

Внимание

Большинство программ выполняет хотя бы минимальную очистку, если они завершаются при выходе пользователя из оболочки GNOME. Но не все программы делают это корректно. Вот почему рекомендуется самостоятельно завершать активные программы перед выходом из системы. Это особенно важно для программ, работающих с файлами, например текстовых процессоров.

Операции с файлами

В стандартной конфигурации оболочки GNOME диспетчер файлов не запускается. Но вообще в распоряжении пользователей есть два диспетчера. Первый из них — это GNU программа Midnight Commander (часть пакета `gnomemc`). Она долгое время входила в состав GNOME, однако в последнее время предпочтение отдается другому диспетчеру: Nautilus (пакет `nautilus`). Если инсталлировать последний, он будет запускаться автоматически.

На с. 100 рассказывалось о том, как использовать программу Nautilus для выполнения основных операций с файлами, таких как копирование, переименование, изменение прав доступа и т.д. Все это не представляет сложностей для пользователей, имеющих опыт работы любой другой в графической оболочке. Особого внимания заслуживает лишь сопоставление типов файлов конкретным приложениям. Но об этом речь пойдет чуть позже.

Конфигурирование рабочего стола

Рабочий стол оболочки GNOME имеет большое число настраиваемых параметров. Можно указать, какие программы должны вызываться из стартового меню, а какие запускаться диспетчером Nautilus при активизации файлов того или иного типа, задать шрифты, используемые в программах GNOME, и многое другое. Основным инструментом настройки является программа Control Center, которая заслуживает более подробного знакомства.

На заметку

В KDE тоже есть программа Control Center, которая имеет сходные функциональные возможности. У оболочки XFce меньше параметров, допускающих настройку, к тому же они "распределены" между несколькими утилитами. Впрочем, все эти утилиты доступны через панель инструментов XFce.

Использование программы Control Center

Для запуска программы Control Center необходимо выбрать в стартовом меню команду Programs⇨Settings⇨GNOME Control Center или выбрать из меню в верхней части экрана команду Settings⇨GNOME Control Center. Можно также ввести `gnomecc` в терминальном окне. На переднем плане рис. 23.2 присутствует окно программы Control Center, в котором активизирован один из модулей. Список модулей приведен в левой панели. Например, чтобы получить доступ к настройкам мыши, следует выбрать модуль Peripherals | Mouse. Если имя модуля известно еще до вызова программы, выберите его непосредственно из меню GNOME Foot⇨Programs⇨Settings или Settings. В этих меню содержатся ссылки на модули программы Control Center, а также на ряд других конфигурационных утилит.

При изменении конфигурационных настроек рекомендуется сначала щелкнуть на кнопке Try, чтобы увидеть эффект от этих изменений, прежде чем делать их постоянными (учтите, что не всякое изменение дает немедленно наблюдаемый эффект). Если результат получился не таким, как ожидалось, щелкните на кнопке Cancel или Revert и все настройки будут восстановлены. В противном случае щелкните на кнопке OK.

Сделав все необходимые установки, выберите в меню File команду Exit или нажмите <Ctrl+C>, чтобы выйти из программы

Установка привязок между типами файлов и программами-обработчиками для диспетчера Nautilus

Привязки между типами файлов и программами-обработчиками устанавливаются с помощью модуля Document Handlers | File Types and Programs программы Control Center (рис 23 4). При выборе этого модуля в окне программы отображается список типов документов. Каждый элемент списка состоит из четырех полей.

- **Описание (Description)** В этом поле указывается тип документа, например *AbiWord document* или *TIFF image*. Если документы этого типа создаются определенной программой, обычно сообщается ее название.
- **Тип MIME (MIME Type)**. Двухсоставной код MIME (Multipurpose Internet Mail Extensions — многоцелевые расширения электронной почты в сети Internet) описывает тип файла. Первая часть кода задает общий класс документа, например текст (*text*) или изображение (*image*), а вторая часть — программу, создающую документ, либо его общий тип, например *abiword* или *tiff*.
- **Расширение (Extension)**. Часть имени файла, стоящая за последней точкой, например .abw или .tif. Диспетчер Nautilus разрешает закреплять за одним типом MIME несколько расширений имен файлов. Чаще всего расширения состоят из трех символов (не считая точки), но в принципе их длина не ограничена. У имени файла может не быть расширения.
- **Действие по умолчанию (Default Action)**. В этом поле указывается, что должно произойти при двойном щелчке на значке файла соответствующего типа в диспетчере Nautilus. Действия по умолчанию закрепляются за типами MIME.

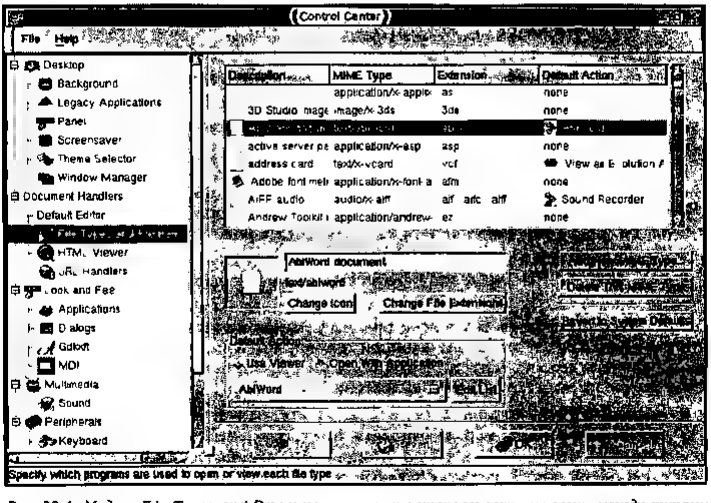


Рис 23 4 Модуль File Types and Programs позволяет устанавливать привязки между типами файлов и программами-обработчиками

Ниже описана последовательность действий которые необходимо выполнить для того, чтобы изменить действие по умолчанию.

1. Выберите нужный тип файла в списке.
2. Установите опцию Open with Application в группе Default Action
3. Раскройте список выбора приложения в группе Default Action Если необходимое приложение присутствует в списке, выберите его и перейдите к п 8
4. Если нужное приложение отсутствует в списке, щелкните на кнопке Edit List Появится диалоговое окно Edit Applications List (рис. 23.5)

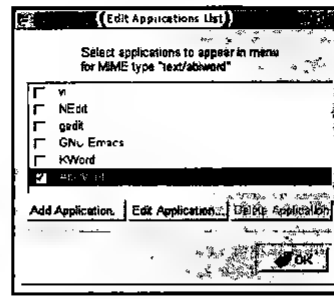


Рис 23 5 В диалоговом окне Edit Applications List приводится список приложений, с помощью которых диспетчер Nautilus считает возможным обрабатывать файлы заданного типа

5. Если нужное приложение есть в списке выберите его и щелкните на кнопке OK после чего вернитесь к п 3.
6. Если же приложения нет в списке, щелкните на кнопке Add Application чтобы отобразить диалоговое окно Add Applications
7. В окне Add Applications задайте имя приложения и укажите полный путь к нему (строка, которую нужно ввести в терминальном окне для запуска приложения), а затем щелкните на кнопке OK Имя приложения появится в списке Edit Applications List (рис 23.5), после чего нужно вернуться к п 5
8. Теперь выбранное приложение ассоциировано с нужным типом MIME Если работа с модулем File Types and Programs на этом завершена щелкните на кнопке OK

Далее можно проверить созданную привязку, найдя в диспетчере Nautilus файл соответствующего типа и выполнив двойной щелчок мыши на нем. Если все было сделано правильно, должна запуститься программа-обработчик и открыть файл В противном случае выполните следующие проверки

- Воспользуйтесь командой ps, чтобы узнать запущено ли интересующее вас приложение. Например, команда `ps ax | grep AbiWord` сообщает о том, работает ли программа AbiWord Попробуйте сделать это несколько раз непосредственно после двойного щелчка на значке файла. Если программа запускается и тут же завершается, перейдите к следующему пункту и попытайтесь выяснить причину сбоя.

- Попробуйте запустить программу в терминальном окне, указав ее имя. Можно также сообщить программе имя файла данных, например `AbiWord myfile.abw` чтобы проверить загрузит ли она файл таким способом. Запуск программы в терминальном окне удобен тем что можно получить полезные сообщения об ошибках.
- Если нет никаких признаков того, что программа запущена, проверьте ин сталлирована ли она и еще раз уточнните имя файла, заданное в п 7 предыду щего перечня

Редактировать существующие файловые привязки нетрудно Несколько более сложная задача — создать совершенно новую привязку Это может понадобиться, если инсталлировано приложение, создающее собственные файлы данных, тип которых не упоминается в программе Control Center. Ниже описана необходимая последователь-ность действий

1. Определите, к какому типу MIME относятся файлы приложения. Как правило, эти сведения есть в документации Попробуйте также выполнить поиск в Internet, указав в качестве ключевых слов тип или расширение файла и фразу "MIME type". Если ничего не помогло, придумайте "правдоподобный" тип, например `text/obiewrite`, для несуществующего текстового процессора Obie Writer.
2. Щелкните на кнопке Add New MIME Type в модуле File Types and Programs (рис 23 4). В результате появится диалоговое окно Add MIME Type в котором можно задать нужный тип MIME и его описание (рис 23 6).

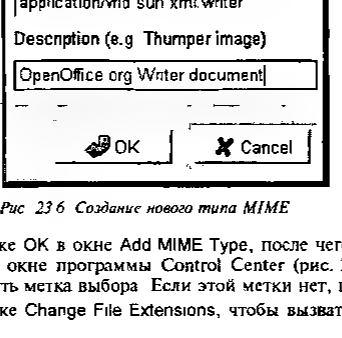


Рис 23 6 Создание нового типа MIME

3. Щелкните на кнопке OK в окне Add MIME Type, после чего запись о новом типе файлов появится в окне программы Control Center (рис. 23 4), причем слева от записи должна стоять метка выбора Если этой метки нет, поставьте ее.
4. Щелкните на кнопке Change File Extensions, чтобы вызвать диалоговое окно File Extensions
5. В окне File Extensions щелкните на кнопке Add Появится диалоговое окно Add New Extension
6. В окне Add New Extension задайте расширения (без указания точки), которые должны быть сопоставлены файлам этого типа Это может быть одно расширение или несколько Во втором случае они разделяются пробелами
7. Последовательно щелкните на кнопках OK в окнах Add New Extension и File Exten sions Теперь заданные расширения должны появиться в списке типов файлов в окне Control Center

8 Выполните описанную ранее процедуру чтобы сопоставить новому типу файлов программу обработчик

Остается протестировать созданную привязку, как это делалось выше Трудность может заключаться в том, что зарегистрированное расширение уже используется другой программой. В этом случае при двойном щелчке на значке файла будет запускаться не то приложение Просмотрите список типов файлов, найдите мешающую привязку и отредактируйте ее: удалите расширение, задайте ссылку на другое приложение или вообще удалите конфликтный тип файла.

Модификация внешнего вида программ

X программы используют оконные интерфейсы, чтобы отображать согласованные наборы элементов управления в окнах Для программиста наличие стандартных оконных интерфейсов существенно упрощает задачу разработки приложений. Вместо того чтобы дублировать строку за строкой, отображая к примеру, кнопки, достаточно вызвать библиотечную функцию вывода кнопки Раньше существовало много оконных интерфейсов, и в каждом из них элементы управления выглядели по-особому Сегодня таких интерфейсов тоже немало, однако многие из них допускают разного рода настройку со стороны пользователей Можно менять цвета и стили кнопок, отображаемых в окнах, стили меню и т.д Чтобы не приходилось делать это в каждой программе, в оболочке GNOME предусмотрены средства глобального конфигурирования определенных параметров пользовательского интерфейса Эти параметры регулируются с помощью модулей программы Control Center

- **Фон рабочего стола** Разрешается настраивать фон, на котором выполняются все программы С помощью модуля Desktop | Background цвет фона можно делать сплошным или градиентным Кроме того в качестве фона можно использовать графический файл.
- **Темы.** С помощью модуля Desktop Theme Selector можно выбирать "тематические наборы" элементов управления Изменение темы в основном затрагивает кнопки, а также фоновые цвета диалоговых окон, шрифты и др.
- **Меню и панели инструментов** С помощью модуля Look and Feel | Applications можно настраивать параметры отображения меню и панелей инструментов, в частности должны ли выводиться подписи к кнопкам панелей где должна отображаться строка состояния и т.д
- **Диалоговые окна** В интерфейсе GTK+ предусмотрен ряд параметров, касающихся размещения кнопок в диалоговых окнах, а также расположения самих диалоговых окон на экране Эти параметры доступны посредством модуля Look and Feel | Dialogs.
- **MDI** Некоторые программы, использующие оконный интерфейс GTK+, соответствуют стандарту MDI (Multiple Document Interface — многодокументный интерфейс), который определяет порядок одновременной обработки нескольких документов одной программой Параметры переключения между документами в таких программах настраиваются с помощью модуля Look and Feel | MDI

На заметку Следует помнить, что ни одна из этих настроек, за исключением параметров фона, не влияет на программы которые не основаны на интерфейсе GTK+ Если вы, к примеру, выбрали тему и обнаружили, что программа не изменила внешний вид, значит, в ней используется другой набор элементов управления Возможно, его параметры тоже можно модифицировать, но только не с помощью программы Control Center

Изменение настроек клавиатуры и мыши

Параметры мыши и клавиатуры задаются с помощью модулей Peripherals | Mouse и Peripherals | Keyboard. Чаще всего подвергаются настройке следующие параметры:

- **Ускорение указателя мыши**. Параметры ускорения задаются с помощью двух регуляторов. Один из них устанавливает порог, т.е. расстояние, на которое должен переместиться указатель, чтобы X-среда начала его ускорение. При работе с графикой, где требуется точное позиционирование указателя, необходимо задавать большой порог. Регулятор ускорения определяет, как быстро растет скорость указателя при непрерывном его перемещении.
- **Конфигурация кнопок мыши**. Параметры Left Handed и Right Handed определяют назначение кнопок мыши. Если вы — левша и привыкли работать с мышью левой рукой, установите опцию Left Handed.
- **Скорость повтора клавиатуры**. Обычно система дублирует символы при удержании клавиши. Разрешается менять длительность задержки перед началом повтора и скорость повтора.
- **Звуковое эхо**. Некоторые пользователи предпочитают слышать звуковое подтверждение при нажатии клавиши. Разрешается включать и отключать режим эха, а также настраивать его громкость.
- **Сигнализация об ошибках**. Некоторые программы выдают звуковой сигнал через динамик компьютера при возникновении определенных ситуаций, например ошибок. Можно менять громкость, тональность и длительность такого сигнала.

Эти параметры являются глобальными и влияют на все программы независимо от используемого оконного интерфейса. Но их действие не распространяется на работу в текстовом режиме (можно нажать <Ctrl+Alt+F1>, чтобы переключиться в текстовый режим), а также на определенные сеансы дистанционной регистрации, например, когда доступ в систему осуществляется посредством VNC-клиента, как описывалось в главе 21. Эти же параметры можно задавать и другими способами, например с помощью программы `xset` или файла `XF86Config`, однако GNOME переопределяет все существующие установки в процессе своего запуска.

Добавление программ в экранные меню

Иногда инсталлируемая программа сама добавляет необходимую ссылку в стартовое меню GNOME. В остальных случаях приходится либо вызывать программу вручную из терминального окна, либо добавлять ссылку на нее в стартовое меню. В последнем случае воспользуйтесь программой Menu Editor (рис. 23.7), которая вызывается с помощью команды GNOME Foot⇨Panel⇨Edit Menus или Program⇨Settings⇨Menu Editor. Можно также ввести `gmenu` в терминальном окне. Учтите, что эта программа не является частью программы Control Center.

Перечень доступных экранных меню появляется в левой панели окна программы. Основных меню три: Favorites, Programs и Applets. Каждое из них может содержать любое число подменю. На рис. 23.7 представлено содержимое меню Programs.

На заметку

Только суперпользователь имеет право редактировать меню Programs и Applets. Они являются общими для всех учетных записей, и именно в них должны добавляться программы, доступные всем пользователям GNOME. Программы, предназначенные для личного использования, добавляются в меню Favorites.

Рис 23.7 Программа Menu Editor позволяет настраивать содержимое экранных меню

Чтобы добавить программу в меню выполните указанные ниже действия

1. В левой панели окна программы Menu Editor выберите меню, где должен появиться новый элемент. На рис 23 7, например, выбрано меню Favorites.
2. Щелкните на кнопке New Item или выберите команду File⇨New Item В меню будет добавлена новая запись, названная Untitled.
3. Введите имя записи в поле Name в правой панели Эта строка будет отображаться в экранном меню
4. При желании можно ввести комментарий в поле Comment. Этот комментарий будет появляться в виде контекстной подсказки при наведении указателя мыши на соответствующий элемент меню
5. В поле Command введите строку вызова программы
6. Задайте тип записи в поле Type В случае программы используйте тип Application Поддерживаются также типы URL (вызов Web-броузера) и Directory (вызов диспетчера Nautilus для соответствующего каталога).
7. Если программа работает в текстовом режиме, установите флажок Run in Terminal В противном случае результаты работы программы нельзя будет увидеть
8. Чтобы задать значок для элемента меню, щелкните на кнопке Icon (по умолчанию она отображает текст No Icon) Появится окно, в котором следует выбрать файл значка. По умолчанию в этом окне отображается содержимое стандартного каталога значков GNOME но можно перейти в любой каталог По окончании щелкните на кнопке OK
9. Щелкните на кнопке Save чтобы сохранить изменения.

Если созданный элемент меню не работает, еще раз проверьте его описание, особенно строку вызова программы, заданную в п 5 Попробуйте запустить программу из терминального окна, чтобы убедиться в ее работоспособности Возможно программа

не инсталлирована или работает некорректно. В терминальном окне могут отобра жаться сообщения об ошибках, позволяющие понять причину сбоя

С помощью программы Menu Editor можно не только добавлять новые программы в меню но и создавать подменю, удалять записи, перемещать их и сортировать Для всех этих функций предусмотрены специальные кнопки в панели инструментов.

Резюме

Настольные оболочки — важная часть современной вычислительной среды Во FreeBSD наиболее распространены две оболочки KDE и GNOME. Более редкая обо лочка — XFce — удобна в системах с ограниченным объемом оперативной памяти и маломощным процессором. Можно даже создать собственную оболочку, если ни одна из доступных оболочек вас не устраивает. FreeBSD позволяет каждому пользователю многопользовательской системы выбрать удобную оболочку, при условии, что их ин сталлировано несколько. Компоненты одной оболочки можно использовать в другой, хотя иногда это приводит к потере части функциональных возможностей Работа в графических оболочках в целом ведется сходным образом В частности, у KDE и GNOME весьма похожие диспетчеры файлов и утилиты конфигурирования рабочего стола

Полный справочник по

Глава 24

Сетевые утилиты

В части IV рассматривались различные сетевые серверы FreeBSD является от личной серверной платформой, но она может также содержать клиентские программы для поддерживаемых серверов. Как правило, клиентские программы про ще конфигурировать, чем серверы. Это позволит в одной главе описать почтовые кли енты, Web-броузеры, FTP-клиенты, средства совместного доступа к файлам и клиен ты дистанционной регистрации. Поскольку у всех клиентов, даже одного класса, есть свои особенности, акцент будет сделан на общих характеристиках клиентов каждого класса.

На заметку

Некоторые клиенты описывались в других главах. Так, в частности, в главе 14 рассказывалось о конфигурировании клиентов DHCP и DNS, NTP-клиенты рассматривались вместе с NTP-серверами в главе 22, кли енты дистанционной регистрации в графическом режиме - в главе 21.

Почтовые клиенты

Серверы электронной почты чрезвычайно широко используются в компьютерных сетях, и FreeBSD поддерживает большое число почтовых клиентов. Для работы с ни ми необходимо иметь доступ к одному или нескольким почтовым серверам. Можно запустить такие серверы на своем компьютере, как описывалось в главе 19, или вос пользоваться сторонними серверами. Основные почтовые операции — это чтение, от правка и сохранение почты. В каждой из программ выполнение перечисленных опе раций имеет свои особенности. Электронные письма часто сопровождаются вложе ниями, посредством которых можно обмениваться нетекстовыми документами, например графическими изображениями и звуковыми файлами. Работая с почтовыми клиентами, необходимо уметь безопасно обрабатывать такие документы.

Наиболее распространенные клиенты

Чтобы иметь возможность читать электронную почту во FreeBSD, необходимо ин сталлировать и сконфигурировать почтовый клиент. FreeBSD поддерживает множест во популярных почтовых программ, от простейших текстовых утилит до современных графических приложений. Привести полный их перечень не представляется возмож ным. Ниже перечислены наиболее популярные клиенты:

- `mail` — простейший текстовый клиент. Он не очень популярен как программа чтения почты, но его удобно вызывать в сценариях автоматизированной от правки почты.
- `pine` (`http://www.washington.edu/pine/`) — довольно сложный текстовый клиент, имеющий удобные средства взаимодействия с пользователем.
- `elm` (`http://www.abo.fi/dc/miniguides/elm-enge.doc.html`) — эта про грамма во многих отношениях аналогична `pine`, но менее удобна для пользо вателей.
- **Mutt** (`www.mutt.org`) — современный текстовый клиент, получивший попу лярность в конце 90-х годов.
- `xmail` — это не самостоятельный почтовый клиент, а компонент редактора Emacs, позволяющий читать в нем почтовые сообщения.
- **Evolution** (`http://www.ximian.com/products/evolution/`) — "ответ сооб щества UNIX на программу Outlook компании Microsoft". Evolution — сложное

графическое приложение, содержащее, помимо почтового модуля календарный модуль и модуль управления информацией.

- **GNUMail.app** (`http://www.collaboration-world.com/gnumail/`) — почтовый клиент, смоделированный на основе программы Mail.app операционных систем NeXTStep и Mac OS X
- **KMail** — часть графической оболочки KDE (K Desktop Environment). Довольно мощный и в то же время небольшой почтовый клиент. Его документация доступна по адресу `http://docs.kde.org/en/3.1/kdenetwork/kmail`
- **Mahogany** (`http://mahogany.sourceforge.net`) — сложная почтовая система с графическим интерфейсом
- **Spruce** (`http://spruce.sourceforge.net`) — графический почтовый клиент небольшой и производительный
- **Sylpheed** (`http://sylpheed.good-day.net`) — небольшой и высокоскоростной почтовый клиент, несмотря на то, что является графическим
- **`xmail`** — X-оболочка для программы `mail`

Большинство программ доступно в разделе `mail` коллекции пакетов или коллекции переносимых приложений. Общие способы инсталляции таких программ рассматривались в главе 11. Программа KMail входит в пакет `kdenetwork`, который относится к разделу `net`, а программа `rmail` инсталлируется вместе с редактором Emacs (раздел `editors`).

Какой из клиентов лучше? Ответ на этот вопрос во многом зависит от личных предпочтений. Некоторые пользователи предпочитают иметь дело с текстовыми клиентами, такими как `mutt`, даже при работе в X-среде. Другим удобнее читать почту в графическом режиме. Некоторые почтовые клиенты, например KMail, тесно взаимосвязаны с другими компонентами настольной среды. Программы типа Evolution имеют множество функциональных возможностей, тогда как утилиты наподобие `mail` намного проще. Попробуйте инсталлировать несколько клиентов и выберите наиболее удобный для себя. Администраторам многопользовательских систем следует инсталлировать набор почтовых клиентов, чтобы у пользователей был выбор.

Почтовые клиенты не всегда используют совместимые форматы почтовых ящиков, поэтому при смене клиента можно потерять доступ к почтовому каталогу. В связи с этим при оценке возможностей почтового клиента рекомендуется создавать тестовую учетную запись.

Начальное конфигурирование

После инсталляции почтовый клиент более-менее готов к работе. Есть лишь несколько параметров, которые могут потребовать модификации. В основном они задаются отдельно по каждому пользователю. Соответствующие конфигурационные файлы находятся в начальных каталогах пользователей. Некоторые программы создают для этих файлов отдельные подкаталоги. Стандартные конфигурационные файлы большинства программ хранятся в каталоге `/etc` или в дереве каталогов конкретной программы. Рассмотрим ряд примеров.

- **`pine`**. Стандартные установки этой программы находятся в файле `/usr/local/etc/pine.conf`. Они могут переопределяться в пользовательских файлах `~/.pinerc`
- **Mutt**. Стандартные установки программы хранятся в файле `/usr/local/etc/Muttrc`. Они могут переопределяться в пользовательских файлах `~/.muttrc`

- **Spruce.** У этой программы нет глобального конфигурационного файла. При первом запуске программа запрашивает значения конфигурационных параметров и записывает их в файл `~/.spruce/spruce.conf`.
- **Sylpheed.** Как и Spruce, программа Sylpheed запрашивает у пользователя стандартные установки во время первого запуска, после чего сохраняет их в начальном каталоге пользователя, в частности в файле `~/.sylpheed/sylpheedrc`.

Некоторые программы — особенно с X-интерфейсом и ряд текстовых клиентов — позволяют пользователям менять установки по умолчанию, не выходя из программы. Если есть возможность, лучше поступать именно так, поскольку это снижает риск повреждения конфигурации из-за ошибки редактирования. На рис. 24.1 приведен пример: диалоговое окно Spruce Configuration Options. Оно появляется при первом запуске программы Spruce и может быть вызвано повторно из меню Settings⇨Preferences. Похожее окно есть и у других графических клиентов.

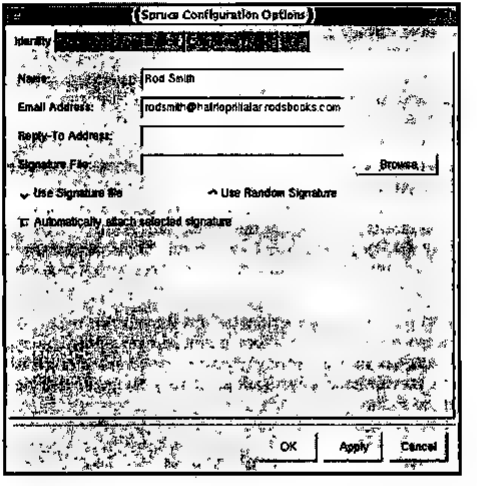


Рис. 24.1. Графические почтовые клиенты упрощают ввод информации об учетной записи

Рассмотрим параметры, которые чаще всего приходится задавать.

- **Имя и адрес.** Многие программы формируют имя пользователя и его почтовый адрес на основании информации из базы учетных записей FreeBSD и стандартных сведений о компьютере, таких как сетевое имя. Эту информацию можно редактировать, например удалять компьютерную часть сетевого имени, оставляя только доменную часть, если ответная почта должна поступать на сервер домена. Многие программы, включая Spruce, даже содержат отдельный параметр для обратного адреса (Reply-To Address).

- **Подпись** – это короткий элемент в конце почтового сообщения, используемый для указания почтового адреса, начальной Web-страницы, названия компании и другой информации. Желательно, чтобы подпись занимала не более 4 6 строк
- **Сервер исходящей почты** Многие почтовые программы FreeBSD (особенно текстовые) по умолчанию направляют исходящую почту программе `sendmail`, работающей на локальном компьютере Другие программы требуют явного указания сервера исходящей почты. Если он функционирует на локальном компьютере, задайте `localhost`
- **Сервер входящей почты**. Почтовая программа должна "знать", где искать входящую почту Это может быть локальный почтовый каталог, если система принимает почту непосредственно или используется программа `fetchmail`, которая принимает почту и помещает ее в локальный каталог (см главу 19) Кроме того, большинство почтовых клиентов можно сконфигурировать на доступ к дистанционному серверу по протоколу POP или IMAP. Клиент может проверять поступление почты с регулярной периодичностью, скажем каждые полчаса В противном случае необходимо вручную выполнять соответствующую команду или перезапускать программу чтобы она проверяла почту
- **Редакторы** Некоторые почтовые клиенты, особенно текстовые, перепоручают задачу редактирования создаваемых сообщений внешним программам К примеру, параметры `set editor` в файле `muttrc` и `editor` в файле `.pinerc` задают программы, используемые в качестве редакторов почты

Большинство почтовых клиентов создает подкаталоги в начальных каталогах пользователей, предназначенные для хранения почты. Они могут называться `Mail`, `mail` или как-то иначе. В почтовом каталоге допускается создавать *папки* Обычно это файлы, содержащие сообщения, но в качестве папок могут выступать и подкаталоги, содержащие по одному файлу на каждое сообщение Если почтовый клиент читает сообщения из локального почтового каталога, то входящая почта доступна и через каталог `/var/mail/`*`имя_пользователя`*

Некоторые почтовые клиенты автоматически пересылают почту из системного почтового каталога в локальную папку, называющуюся `inbox` или как-то иначе Это усложняет восстановление почты в ситуации, когда тестируется почтовый клиент помещающий почту в свою собственную папку

Чтение и хранение электронной почты

Большинство клиентов разбивает сообщения по папкам Способ доступа к этим папкам зависит от программы Текстовые программы, такие как Mutt и `pine`, используют команды, которые можно вводить для "перемещения" в папку. После этого можно получить список сообщений в папке, ввести для просмотра сообщения его номер и т.д. Графические программы выводят список почтовых папок в окне или в панели главного окна либо содержат кнопку, при нажатии которой выводится список папок. Например, на рис. 24 2 представлено окно программы Spruce В левой верхней панели отображаются все доступные папки, разбитые на три категории *Local Mailboxes* (локальные почтовые ящики, которые принадлежат программе Spruce и хранятся в каталоге `~/.spruce`), *Remote Mailboxes* (почтовые ящики, поддерживаемые сервером IMAP, с которым взаимодействует программа Spruce) и *Local Spool* (почтовые ящики которые могут использоваться совместно с другими почтовыми клиентами включая основной буфер входящей почты). Не все почтовые клиенты ор

ганизуют почтовые ящики таким способом, но подобная организация типична В каждой категории представлены отдельные папки например *Inbox* (входящие), *Drafts* (черновики) и др

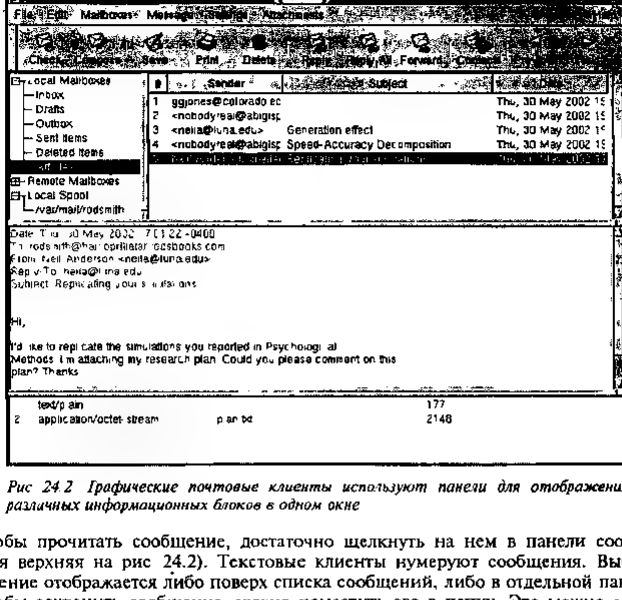


Рис 24 2 Графические почтовые клиенты используют панели для отображения различных информационных блоков в одном окне

Чтобы прочитать сообщение, достаточно щелкнуть на нем в панели сообщений (правая верхняя на рис 24.2). Текстовые клиенты нумеруют сообщения. Выбранное сообщение отображается либо поверх списка сообщений, либо в отдельной панели

Чтобы сохранить сообщение, нужно поместить его в папку Это можно сделать с помощью соответствующей команды меню или перетаскиванием. Большинство клиентов позволяет создавать новые папки. Например, папка *Articles* на рис. 24.2 не является стандартной частью почтового каталога программы Spruce.

Иногда возникает необходимость сохранить сообщение в отдельном файле. Обычно сохраненная версия сообщения содержит минимальный набор *заголовков*, т е информационных "шапок", используемых почтовыми серверами при доставке сообщения получателю и для записи сведений о системах, через которые проходит сообщение на пути к адресату.

Совет Чтобы обойти это ограничение, создайте новую почтовую папку и переместите сообщение в нее Полученный файл будет содержать одно сообщение со всеми заголовками

Отправка электронной почты

Когда пользователь решает составить сообщение, почтовый клиент запускает внешний редактор, такой как `jed`, `pico` или `vi`, либо отображает собственное окно редактирования Как правило, это очень простой редактор с минимальным набором

возможностей. В основном пользователи электронной почты обмениваются обычными текстовыми сообщениями, поэтому почтовые редакторы почти не содержат средств форматирования. Лишь в последние годы HTML, использовавшийся ранее при создании Web-страниц, приобрел популярность как язык форматирования почтовых сообщений. Почтовые редакторы, поддерживающие язык HTML, могут создавать различные текстовые эффекты, таблицы и даже встроенную графику.

Не все пользователи любят работать с сообщениями формата HTML. Такие сообщения обычно имеют больший размер (особенно при использовании встроенной графики) и они дольше загружаются по медленным коммутируемым линиям. Если почтовый клиент получателя не умеет анализировать HTML-сообщения, пользователь увидит исходные коды разметки текста, что затруднит чтение сообщения.

Когда пользователь сохраняет сообщение и выходит из редактора или нажимает кнопку Send, почтовый клиент делает попытку отправить сообщение. Некоторые клиенты (особенно текстовые) по умолчанию вызывают локальную программу `/usr/sbin/sendmail`, которая прекрасно работает во FreeBSD, хотя при этом нужно помнить о некоторых нюансах. В частности, ряд провайдеров запрещает непосредственную отправку почты (стандартный режим работы программы `sendmail`). Кроме того, если система-получатель не отвечает немедленно, программа `sendmail` через определенное время повторяет попытку. Такой подход удобен, когда Internet-соединение функционирует постоянно. В главе 19 описывалось, как сконфигурировать `sendmail` на использование ретранслятора исходящей почты, что позволяет преодолеть указанные проблемы. В качестве ретранслятора можно задать почтовый сервер провайдера.

Вложения

Вложения — чрезвычайно важные компоненты электронной почты. Это способ присоединения файлов к почтовым сообщениям в соответствии со стандартом MIME (Multipurpose Internet Mail Extensions — многоцелевые расширения электронной почты в сети Internet). В виде вложений можно пересылать текстовые документы, изображения, звуковые, архивные и другие файлы (вообще говоря, HTML-сообщения реализуются именно как вложения: HTML-документ посылается в виде дополнения к пустому письму или к текстовой версии того же сообщения). Когда приходит письмо, почтовый клиент указывает на то, что имеется вложение. В программе Sprnce (рис. 24.2) информация о вложении появляется в нижней панели. Если щелкнуть на обозначении вложения, присоединенный файл будет загружен в соответствующую программу обработки или же появится меню со списком доступных комаид.

В последние годы почтовые вложения стали широко использоваться для распространения вирусов. Большинство атак такого рода нацелено на Windows-системы, но теоретически почтовые "черви" и другие злонамеренные программы способны атаковать и UNIX-системы. Никогда не открывайте вложение, если вы не уверены в его безопасности.

Большинство почтовых клиентов поддерживает возможность отправки вложений. У графического клиента для этого предусмотрена специальная кнопка с изображением скрепки или команда меню, при выборе которой появляется окно диспетчера файлов.

Учитывайте размеры файлов, присоединяемых к почтовым сообщениям. Большинство провайдеров ограничивает размер обрабатываемых сообщений, поэтому письма со слишком объемными вложениями не достигнут адресата. Но даже если письмо дойдет до получателя, вряд ли он будет в восторге от перспективы загружать многомегабайтовое

сообщение по медленному коммутируемому каналу Правилами хорошего тона предусмотрено получение разрешения на отправку сообщения размером более 1 Мбайт

Пользователи часто не отдают себе отчет в том, что размер почтового сообщения может оказаться больше, чем размер исходного файла Это следствие того, что почтовые клиенты кодируют бинарные вложения, используя протокол, рассчитанный на текстовые данные. Для кодирования одного байта простого текста требуется шесть битов, тогда как в бинарных файлах значащими являются все восемь битов байта. Стандарты MIME-вложений решают эту проблему за счет добавления байтов. Например, для кодирования шести байтов бинарных данных (48 бит) может понадобиться восемь 6-битовых символов К ним еще добавляются управляющие символы В результате бинарные вложения занимают на 35% больше полосы пропускания, чем необходимо для передачи тех же файлов другим способом, например через FTP-сервер. С другой стороны, почтовые вложения очень удобны, особенно если ни отправитель, ни получатель не имеют доступа к FTP-серверу.

Перед отправкой больших бинарных файлов постарайтесь сжать их с помощью утилиты `gzip` или `zip` Конечно, 35-процентная "надбавка" все равно будет иметь место, но сжатие может перекрыть этот эффект

Web-броузеры

Web-броузеры применяются практически повсеместно и большинство пользователей знают, как с ними работать. По этой причине мы не будем слишком углубляться в детали функционирования броузеров, сделав акцент на обзоре броузеров, доступных во FreeBSD их конфигурировании и безопасности

Наиболее распространенные броузеры

В настоящее время наиболее популярен броузер Internet Explorer компании Microsoft, однако он недоступен во FreeBSD Это создает определенные неудобства при просмотре Web-страниц, разработчики которых не потрудились учесть ограничения Internet Explorer Впрочем, большинство Web-страниц вполне нормально отображается многочисленными броузерами FreeBSD

- **Netscape Navigator** (`www.netscape.com`). Ранее это был основной конкурент броузера Internet Explorer, но за последние несколько лет его рыночная доля резко сократилась К сожалению, лишь очень старые версии Netscape доступны в "родном" формате FreeBSD, да и они имеют проблемы, связанные с безопасностью Можно запустить более новые Linux-версии Netscape, воспользовавшись режимом бинарной совместимости с Linux, как описывалось в главе 4
- **Mozilla** (`www.mozilla.org`). В начале 1998 года компания Netscape объявила о планах выпуска броузера Navigator на условиях открытой лицензии, т е. с открытым исходным кодом. Эти планы в конце концов привели к разработке броузера Mozilla, являющегося открытым эквивалентом Netscape Navigator Они работают сходным образом, однако броузер Mozilla доступен в формате, адаптированном к особенностям FreeBSD, поэтому в данной операционной системе ему отдается предпочтение
- **Konqueror** — является частью проекта KDE и заодно служит диспетчером файлов этой графической оболочки В сравнении с Netscape Navigator и Mozilla

броузер Konqueror выглядит небольшим и быстродействующим, хотя он и не обладает таким набором функциональных возможностей. Броузер инсталлируется вместе с пакетом kdebase, а его документация доступна по адресу http://docs.kde.org/en/3.1/kdebase/konqueror/.

- **Amaya** (http://www.w3.org/Amaya/). Этот Web-броузер довольно необычен тем, что служит "полигоном" для испытания множества новых Web-технологий. В нем интегрирован WYSIWYG-редактор Web-страниц.
- **Emacs**. Его называют "интегрированным" редактором, поскольку он содержит множество компонентов, нехарактерных для "простого" редактора. Сюда входит и Web-броузер, позволяющий фанатам Emacs просматривать Web-страницы, не выходя из редактора. Этот Web-броузер инсталлируется как отдельный добавочный пакет emacs-w3m-emacs21, находящийся в разделе www коллекции пакетов или коллекции переносимых приложений.
- **Galeon** (http://galeon.sourceforge.net). Одним из следствий того, что компания Netscape открыла код броузера Navigator, стала возможность использования ядра этого кода (механизм HTML-визуализации) для разработки новых броузеров. Так появился Galeon, являющийся в настоящее время частью проекта GNOME (GNU Network Object Model Environment).
- **Opera** (www.opera.com) — один из немногих коммерческих Web-броузеров во FreeBSD. Он, впрочем, не является приложением FreeBSD, так как разрабатывался для Linux, поэтому для его работы нужны средства совместимости с Linux, описанные в главе 4. За пределами мира UNIX броузер Opera является ближайшим конкурентом броузеров Internet Explorer и Netscape. Он инсталлируется в виде пакета linux-opera, находящегося в разделе www коллекции пакетов или коллекции переносимых приложений.
- **Lynx** (http://lynx.isc.org/current/). Это популярный Web-броузер, работающий в текстовом режиме. Он позволяет просматривать Web-страницы даже в отсутствие X-среды. Это также важное средство разработки Web-страниц, особенно с точки зрения оценки их доступности. Если Web-страница выглядит четко и понятно в Lynx, то она будет удобна для людей с нарушениями зрения, которые используют технологию синтеза речи.

Это далеко не полный список. Став доступным, исходный код Netscape Navigator положил начало многим проектам, не только Mozilla. Иногда кажется, что похожие броузеры появляются чуть ли не каждую неделю. Большинство из них откровенно "сырые": они содержат ошибки, медлительны, лишены важных функциональных возможностей. Наиболее "зрелые" и полезные броузеры для FreeBSD — Mozilla, Galeon, Konqueror и, с натяжкой, Opera (учитывая, что это Linux-приложение). Желательно инсталлировать хотя бы два из них. Тогда, если вам встретится Web-страница, некорректно отображаемая в основном броузере, можно будет прибегнуть к помощи второго. Поскольку Mozilla и Galeon имеют общий механизм визуализации, рекомендуется в дополнение к одному из них инсталлировать Konqueror или Opera.

Конфигурирование броузера

Броузер готов к работе сразу после инсталляции. В ходе начального запуска некоторые броузеры запрашивают дополнительную информацию, например почтовый адрес пользователя. Такие броузеры обычно содержат интегрированные почтовые клиенты, которые позволяют отправлять почту при щелчке на ссылке mailto: имеющейся на Web-странице.

У любого Web-броузера есть диалоговое окно с большим количеством конфигурационных настроек. На рис. 24.3 представлено окно установок броузера Mozilla, вызываемое из меню Edit⇨Preferences. В левой панели выбирается категория настроек. Укажем ряд параметров, которые могут потребовать изменения.

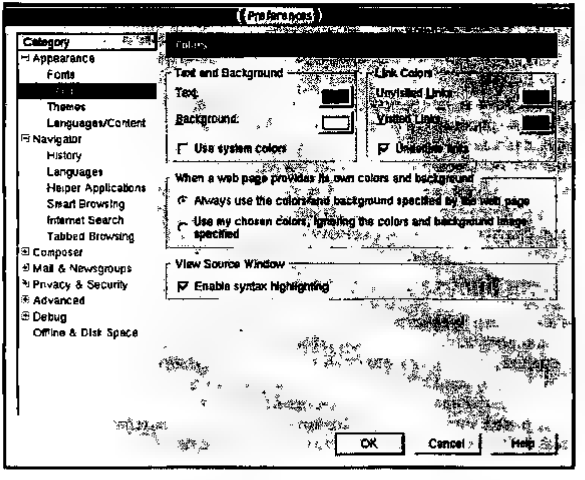


Рис. 24.3. Окно настроек Web-броузера

- **Идентификация.** Броузер позволяет пользователю задать информацию о себе: имя, почтовый адрес и т.д. Эти сведения применяются почтовым модулем. Кроме того, почтовый адрес может послужить паролем при доступе к FTP-узлу в анонимном режиме.
- **Цвета.** Задайте цвета для текста и фона, соответствующие вашему вкусу. Некоторые броузеры позволяют переопределять цвета, установленные в документе, что дает возможность улучшить вид плохо спроектированных Web-страниц.
- **Шрифты.** Обычно шрифты, используемые броузером для конкретных видов текста, можно менять. В некоторых случаях проблемы возникают с размером шрифта. Методы установки размеров шрифта на Web-страницах иногда приводят к тому, что текст отображается слишком большим или, наоборот, маленьким шрифтом. Во избежание этого в броузерах предусматривают минимальный и максимальный кегль.
- **Графика.** Большинство броузеров автоматически загружает и выводит все графические изображения. Этот режим можно отключить, чтобы ускорить отображение Web-страниц.
- **Файлы "cookie".** Файлом "cookie" называется небольшой информационный фрагмент, передаваемый Web-броузеру сервером и возвращаемый Web-броузером по запросу. Такие файлы служат для контроля деятельности пользователей, как в легитимных целях (идентификация клиента при осуществлении интерактивной покупки), так и в более сомнительных случаях (анализ дейст

вий пользователя с точки зрения маркетинга с последующим "навязыванием" рекламы). Правила обработки броузером файлов "cookie" можно изменить, но в основном приходится разрешать их, так как иначе броузер слишком часто выдает запросы вида "Принимать файл?", что замедляет просмотр страниц или же некоторые страницы начинают отображаться некорректно

- **Прокси-серверы.** Часто в сети функционирует прокси-сервер, через который проходят запросы к Web-страницам. Он представляет собой высокоуровневый брандмауэр. В настройках броузера можно задать адрес прокси-сервера и номер его порта
- **Сценарии.** Java и JavaScript — это языки программирования, позволяющие разработчикам расширять возможности Web-страниц. К сожалению написанные на этих языках программы и сценарии не лишены слабых мест с точки зрения безопасности Особенно это касается Java
- **Пользовательский интерфейс** Многие броузеры содержат дополнительные панели, могут отображать несколько окон и т.д Соответствующие настройки обычно содержатся в окне конфигурирования

Безопасность броузера

В идеале просмотр Web-страниц не должен угрожать безопасности компьютера. К сожалению, наш мир далек от идеала Популярность сети Internet делает ее далеко не безопасной средой, поскольку злоумышленники постоянно ищут и используют слабые места в Web-броузерах. Другая неприятная (хотя и не столь опасная) особенность заключается в том, что владельцы Web-страниц могут отслеживать действия пользователей, собирая о них информацию, которую сами пользователи предпочли бы не раскрывать. Следует учитывать такие аспекты безопасности

- **Разглашение ключевой информации.** С каждым HTTP-запросом (HTTP — это протокол, лежащий в основе World Wide Web) броузер предоставляет информацию о себе, ключевые элементы которой — IP-адрес, имя броузера и операционная система компьютера Некоторые броузеры можно сконфигурировать на передачу заведомо ложных сведений, однако IP-адрес невозможно исказить, за исключением случаев, когда в сети используется прокси-сервер или механизм NAT (Network Address Translation — трансляция сетевых адресов). Если кто то прослушивает соединение с помощью анализатора пакетов, он сможет многое узнать о вас проето выяснив какие Web-страницы вы чаще всего посещаете
- **Ошибки броузера.** Ошибки в программном коде могут вызвать сбой броузера но главное, они способны привести к отключению защитных механизмов или дать возможность внешнему пользователю получить доступ к локальным файлам. Старайтесь своевременно обновлять броузер и следите за сообщениями о его безопасности
- **Сценарии** Как указывалось выше, языки Java и JavaScript могут быть источниками проблем, поскольку они позволяют несанкционированным пользователям выполнять произвольный код на компьютере броузера В идеале их нужно отключить, однако это приведет к тому, что большое число Web-страниц будет отображаться некорректно, поэтому какие то послабления неизбежно приходится делать
- **Файлы "cookie"** Файлы cookie позволяют разработчикам Web-узлов следить за действиями пользователей Теоретически содержимое файла "cookie" известно только одному компьютеру однако бывает что разработчики Web-

узлов, связанные деловыми отношениями, делятся сведениями друг с другом, к тому же информация, содержащаяся в файлах "cookie", может послужить для навязывания разного рода рекламы Можно отключить обработку файлов "cookie" в броузере или воспользоваться прокси-сервером, защищающим конфиденциальность пользовательских данных, например Privoxy (`http://www.privoxy.org`)

- **Безопасные транзакции.** Передаваемая Web-броузером на сервер информация может быть перехвачена взломщиками Вот почему необходимо защищать конфиденциальные данные, такие как пароли, номера кредитных карточек и параметры банковских счетов Современные броузеры поддерживают SSL (Secure Sockets Layer — протокол безопасных сокетов), который обеспечивает шифрование передаваемых данных (в будущем взлом протокола SSL может упроститься, но в настоящее время он вполне безопасен, особенно его 128 разрядные версии) Переходя в защищенный режим, броузеры сигнализируют об этом в строке состояния, например, специальным значком с изображением замка Этот значок должен присутствовать при передаче конфиденциальной информации
- **Пароли** Для доступа ко многим Web-узлам требуются пароли Высылать эти пароли через незащищенное соединение слишком опасно К сожалению, далеко не все Web-узлы обеспечивают защиту паролей Кроме того, не так-то легко запоминать многочисленные пароли доступа к различным Web-узлам Некоторые броузеры могут запоминать используемые пароли, однако это тоже имеет определенный риск, ведь в таком случае пароли хранятся на жестком диске откуда их можно украсть В любом случае рекомендуется использовать уникальные пароли для доступа к особо важным Web-узлам, таким как интерактивные банковские службы
- **Ненадежные файлы** Большинство файлов, доступных в Internet, создано людьми, намерения которых были самыми искренними, однако всегда существует вероятность встретить программу с заманчивым описанием, которая на деле оказывается *троянским конем*, т е ее истинное назначение весьма неблаговидно Всегда проверяйте надежность загружаемых программ прежде чем запускать их у себя

Чаще всего пользователи сталкиваются с нарушениями конфиденциальности, которые происходят из-за файлов "cookie" Но гораздо большую опасность несут сценарии Java (в меньшей степени — JavaScript), троянские кони и ошибки броузера Безопасность Web-броузера можно повысить, используя прокси-серверы и корректно настраивая ряд ключевых параметров броузера Следите также за тем, чтобы пароли номера кредитных карточек и другие конфиденциальные данные не передавались по незашифрованным каналам.

Клиенты FTP и SFTP

Протокол FTP существует уже очень давно и печально известен проблемами, связанными с передачей незашифрованных паролей, но это не уменьшает его популярность В некоторых случаях, например при анонимном доступе, безопасность пароля не играет роли Как средство открытого распространения файлов, протокол FTP имеет преимущества перед HTTP, поскольку FTP клиенты часто могут восстанавливать прерванные операции Например, если вы загрузили 630 из 640 Мбайт образа компакт-диска FreeBSD, после чего соединение разорвалось, то в случае протокола HTTP файл придется загружать заново А в протоколе FTP после восстановления соедине

ния можно попросить клиент возобновить пересылку что позволит "докачать" лишь последние 10 Мбайт.

Существует большое число FTP-клиентов. Одни из них являются текстовыми программами, другие имеют графический интерфейс. Ниже будут описаны популярные FTP-клиенты и поддерживаемые ими команды для загрузки файлов как с общедоступных FTP-серверов так и с частных узлов для доступа к которым требуется пароль.

Сервер SSH (Secure Shell) также располагает средствами пересылки файлов, включая защищенную разновидность протокола FTP: SFTP Некоторые FTP-клиенты умеют взаимодействовать с серверами SFTP В этом случае используются те же самые команды что и при подклю чении к традиционным FTP-серверам, поэтому приводимый ниже ма терпал справедлив и для SFTP.

Наиболее распространенные FTP-клиенты

Учитывая возраст и популярность протокола FTP, неудивительно, что во FreeBSD существует большое число FTP-клиентов Перечислим наиболее популярные из них

- **ftp** — исходный текстовый FTP-клиент, инсталлируемый вместе с базовым набором программ FreeBSD и потому всегда доступный Версия клиента во FreeBSD содержит ряд расширений, позаимствованных у более сложных про грамм, таких как ncftp.
- **ncftp** (www.ncftp.com) — текстовая программа, напоминающая исходную утилиту ftp, но с расширенными возможностями, например индикатором за грузки FreeBSD-версия ftp вобрала в себя многие из этих расширений, по этому различия между программами во FreeBSD во многом стерты
- **Mftp** — FTP-клиент работающий в X-среде (содержится в пакете mxftp). По сравнению с другими графическими клиентами пользовательский интерфейс Mftp гораздо проще.
- **AxY FTP** (www.wxftp.seul.org) — графический FTP клиент, пользовательский интерфейс которого смоделирован на основе популярной Windows-программы WS FTP.
- **gFTP** (http://gftp.seul.org) — популярный графический клиент, особенно в системах, где используется оболочка GNOME Пользовательский интерфейс клиента напоминает интерфейс программ AxY FTP и WS FTP. Особенностью клиента является поддержка протокола SFTP, реализуемого серверами SSH
- **hsftp** (http://la-samhna.de/hsftp/). С технической точки зрения это не FTP-клиент, а текстовая программа, которая реализует FTP-подобный набор команд для доступа к файлам на сервере SSH. Таким образом, ее можно использовать в качестве безопасной замены стандартному FTP-клиенту, если в удаленной системе работает сервер SSH вместо сервера FTP (или в дополнение к нему)
- **Web-броузеры.** Все современные Web-броузеры поддерживают протокол FTP Чтобы получить доступ к FTP-узлу, достаточно ввести URL-адрес, начинаю щийся с ftp://, а не с http:// или щелкнуть на соответствующей ссылке Web-страницы. Впрочем Web-броузеры реализуют не все средства FTP, по этому если задача не ограничивается загрузкой нескольких файлов с FTP сервера в анонимном режиме, проще использовать специализированный FTP клиент.

За исключением утилиты ftp, являющейся стандартным компонентом FreeBSD, и некоторых Web-броузеров, все перечисленные программы доступны в виде пакетов или переносимых приложений. Таким образом, к ним можно применять методы инсталляции, описанные в главе 11. В многопользовательской среде желательно установить несколько клиентов, чтобы у пользователей был выбор.

Помимо традиционных FTP-клиентов существует ряд более специализированных программ, использующих протокол FTP. Некоторые из них предназначены для создания *зеркальных копий* FTP-узлов, т.е. для копирования всех файлов FTP-узла на локальный компьютер. Таким способом можно, например, скопировать FTP-узел FreeBSD, чтобы упростить инсталляцию системы в сети. Есть инструменты Web-разработки, позволяющие передавать файлы непосредственно на Web-сервер по протоколу FTP. Различные FTP-клиенты доступны в разделе ftp коллекции пакетов или коллекции переносимых приложений FreeBSD.

Базовые FTP-команды

Протокол FTP определяет набор операций, таких как отправка файла, загрузка файла и т.д. При работе с текстовым FTP-клиентом все эти операции реализуются в командной строке. В случае графического клиента вводимые команды заменяются командами меню, кнопками и т.д. Ниже приведен список наиболее важных команд используемых утилитой ftp и другими текстовыми FTP-клиентами.

- **ascii**. Одно из достоинств протокола FTP заключается в поддержке двух режимов передачи файлов *бинарного*, при использовании которого файлы не подвергаются модификации, и *текстового* (ASCII режим), когда символы конца строки, а иногда и другие символы, модифицируются в соответствии со стандартами принимающей системы. Команда ascii включает текстовый режим.
- **binary**. Как нетрудно догадаться, эта команда включает бинарный режим. Как правило, именно он и установлен по умолчанию, хотя это нужно проверять. Бинарный режим нужно включать перед отправкой нетекстовых файлов, иначе они будут повреждены.
- **bye**, **exit**, **quit**. Эти команды завершают FTP-сеанс и позволяют выйти из программы.
- **cd** — аналог одноименной команды интерпретаторов FreeBSD. Она осуществляет смену рабочего каталога, но на компьютере FTP-сервера. Аргументом команды является путь к каталогу (полный или сокращенный).
- **delete** — удаляет заданный файл на сервере.
- **dir**, **ls** — команды, отображающие содержимое заданного каталога на сервере или текущего каталога, если аргумент отсутствует. На некоторых серверах команда dir формирует более длинный листинг (эквивалент команды ls -l интерпретатора), чем ls.
- **ftp**, **open** — команды, начинающие новый сеанс FTP-доступа. В качестве аргумента задается имя компьютера, например **open ftp.threeroomco.com**.
- **get** — одна из наиболее важных команд протокола FTP: она загружает указанный файл с сервера. Можно задать альтернативное имя, которое будет присвоено сохраняемому файлу.
- **help**, **?** – выводят сводку команд, поддерживаемых FTP-клиентом. Если в качестве аргумента указать имя команды, будет выдано ее краткое описание.
- **lcd**. Команды, начинающиеся с префикса 'l', функционируют *локально*, т.е. в клиентской системе. Команда lcd меняет локальный рабочий каталог, в кото-

ром сохраняются загружаемые файлы и из которого берутся отправляемые файлы

- **lpwd** — отображает имя текущего рабочего каталога клиентского компьютера
- **mget** — загружает группу файлов с сервера. Можно перечислить все файлы по именам или воспользоваться метасимволами, аналогичными подстановочным знакам интерпретатора команд FreeBSD. В зависимости от значения параметра `prompt` может выдаваться запрос на подтверждение каждой операции
- **mkdir** — создает каталог на сервере.
- **mput** — копирует группу файлов на сервер. Как и в случае команды `mget` можно задавать файлы поименно или использовать метасимволы
- **prompt** — разрешает (`on`) или запрещает (`off`) выдачу подтверждений по каждому файлу в групповой операции (команды `mget` и `mput`)
- **put** — антипод команды `get`, копирует одиночный файл с клиентского компьютера на сервер.
- **pwd** — сообщает имя рабочего каталога на сервере
- **reget**. Эта команда полезна, если передача была прервана, она заставляет систему возобновить ее, начиная с последнего байта файла, существующего в локальной системе. Это позволяет избежать повторной пересылки существующих данных
- **rename** — переименовывает файл на сервере. В качестве аргументов указываются текущее и новое имя.
- **rmdir** — удаляет каталог на сервере

Это неполный список, но перечисленных команд вполне достаточно, чтобы начать работу с утилитой `ftp` или другим текстовым клиентом, например `ncftp`. Ниже будет продемонстрировано применение ряда команд.

Графические FTP-клиенты реализуют в целом те же функции, что и их текстовые аналоги. Рассмотрим рис. 24.4, на котором представлено окно программы gFTP. Основные компоненты здесь — панели со списками файлов. Правый список соответствует удаленной системе; он пуст до тех пор, пока не установлено соединение с сервером. Используйте кнопки со стрелками, расположенные между панелями, для копирования выбранных файлов в соответствующем направлении. Команды переключения режима находятся в меню FTP. Меню Local и Remote предназначены для применения команд к файлам локальной или удаленной системы.

Применение FTP: передача файлов

Для передачи файлов по протоколу FTP нужно установить соединение с сервером, найти нужные файлы, ввести команды копирования, а затем завершить сеанс. Конкретные детали зависят от клиента, но базовые этапы остаются неизменными. Ниже описываются загрузка бинарного файла `sample.tgz` с фиктивного FTP-узла (`ftp.threeroomco.com`) и копирование текстового файла `report.txt` на этот же узел с использованием программ `ftp` и gFTP.

Утилита ftp

Поскольку утилита `ftp` всегда доступна, сначала рассмотрим ее применение. Искомая процедура выглядит так.

1. Перейдите в каталог, куда нужно поместить загружаемый файл (`sample.tgz`).

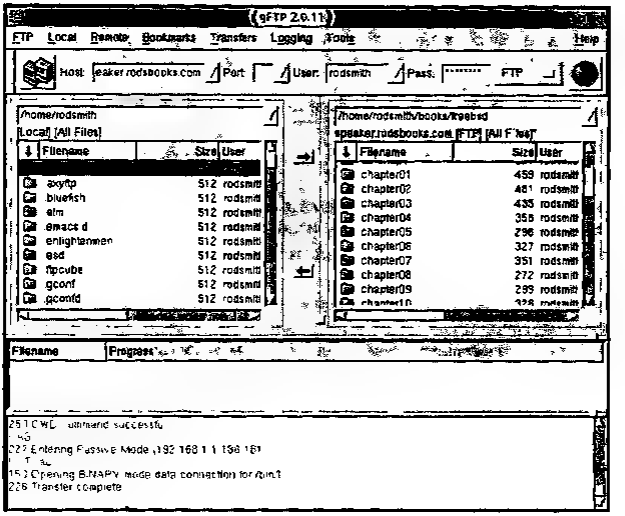


Рис. 24.4. Графические FTP-клиенты выводят списки файлов локальной и удаленной систем, которые удобно просматривать и обрабатывать

2. Введите **ftp ftp.threeroomco.com**. Эта команда установит соединение с FTP-сервером.
3. Сервер выдаст запрос на ввод имени пользователя. Укажите имя учетной записи на сервере или **anonymous**, если возможен анонимный доступ.
4. Сервер потребует ввода пароля. Укажите пароль или адрес электронной почты в случае анонимного доступа.
5. Воспользуйтесь командой `cd` для перехода в каталог, где находится файл `sample.tgz`, например **cd tarballs/samples**. Если точное имя каталога неизвестно, поищите его с помощью команды `ls`. На некоторых серверах есть файлы с именем вида `index`, `00index` и `tn`. В них содержится информация обо всех файлах FTP-сервера.
6. Введите **binary**, чтобы включить бинарный режим пересылки файлов.
7. Введите **get sample.tgz**, чтобы загрузить файл. Появится информация о ходе загрузки, после чего будет выдана сводка о копировании с указанием его скорости и размера файла.
8. Воспользуйтесь командой `cd`, чтобы перейти в каталог сервера, куда будет записан файл `report.txt`, например **cd ~/uploads**.
9. С помощью команды `lcd` перейдите в локальный каталог, где находится файл `report.txt`, например **lcd ~/reports**.
10. Введите **ascii**, чтобы включить текстовый режим пересылки файлов.

11 Введите **put report txt** чтобы скопировать файл на сервер Как и в случае загрузки файла, появится индикатор выполнения Копирование не разрешается при анонимном доступе а также в том случае, когда у пользователя нет права записи в выбранный каталог.

12 Введите **quit**, чтобы разорвать соединение с сервером и выйти из программы `ftp`.

Программа gFTP

Программа gFTP является популярным графическим FTP-клиентом FreeBSD Вот как с ее помощью осуществляется пересылка файлов `sample.tgz` и `report.txt`

1 Перейдите в каталог, куда нужно поместить загружаемый фаил (`sample.tgz`)

2 Введите **gftp**, чтобы запустить программу. Появится окно, аналогичное изображенному на рис. 24 4 но некоторые поля будут пустыми включел список файлов в правой панели.

3 Введите **ftp threeroomco com** в поле Host имя пользователя или **anonymous** в поле User, пароль или адрес электронной почты в поле Pass и нажмите <Enter> В правой панели должен появиться список файлов и подкаталогов находящихся в рабочем каталоге сервера

4 Найдите в правом списке каталог, содержащий файл `sample.tgz`

5 Перейдите в меню FTP и убедитесь в том, что флаг FTP⇨Binary установлен

6 Щелкните на имени файла `sample.tgz` в правой панели

7 Щелкните на кнопке со стрелкой влево. Это инициирует передачу файла В нижней панели появятся сведения о копировании, включая скорость передачи и размер файла

8 Найдите в левой панели каталог клиентской системы, в котором находится файл `report.txt`, а в правой панели — каталог, куда должен быть помещен этот файл Установите флаг FTP ⇨ Ascii, чтобы включить текстовый режим передачи файла

9 Щелкните на имени файла `report.txt` в левой панели.

10 Щелкните на кнопке со стрелкой вправо. Это инициирует передачу файла В нижней панели появится отчет об операции копирования

11 Выберите команду FTP⇨Quit или нажмите <Ctrl+Q> чтобы выити из программы

Клиенты совместного доступа к фвйлам

Серверы передачи файлов, такие как FTP и серверы совместного доступа к файлам, например NFS и Samba, имеют много общего. Даже их клиенты похожи. Например, клиент Samba `smbclient` весьма напоминает утилиту `ftp`. Впрочем, большинство клиентов совместного доступа к файлам — это компоненты операционной системы Они позволяют обычным программам работать с файловыми серверами так как если бы это были локальные жесткие диски

Экспортируемые каталоги NFS

В сетях UNIX широко применяется протокол NFS Пользователи FreeBSD могут легко получить доступ к файлам, хранящимся на сервере NFS

Экспортируемый каталог NFS (т е каталог, который доступен на сервере) монти руется так же, как и локальный раздел или съемный диск Если это нужно сделать од норазово, воспользуйтесь командой mount

```
# mount -t nfs сервер /точка_экспорта точка_монтирования
```

Здесь *сервер* — это имя компьютера, на котором работает NFS-сервер, *точ ка_экспорта* — имя экспортируемого каталога, а *точка_монтирования* — имя локаль ного каталога, к которому подключается каталог сервера Поддерживаются стандарт ные параметры монтирования, в частности r (монтирование только для чтения). Подробнее команда mount описывалась в главе 7

Другой способ монтирования реализуется с помощью команды mount nfs Она работает аналогично mount, но поддерживает специализированные параметры прото кола NFS Рассмотрим наиболее важные из них

- 2 Обычно FreeBSD пытается использовать протокол NFS 3, а если он не ра ботает, переходит на NFS 2 Описываемая опция включает принулительное использование версии 2 что ускоряет работу серверов NFS 2 и делает ее более устойчивой
- 3 — включает принудительное использование версии 3 протокола NFS Это может понадобиться в том случае, если некоторые особенности протокола NFS 3 столь важны что в их отсутствие соединение вообще не должно уста навливаться
- D *порог* заставляет систему выдавать сообщение об ошибке по достижении заданного числа тайм аутов в ходе подключения к серверу Она полезна, если сетевое соединение или сервер ненадежны и локальные программы начинают "зависать"
- R *число_попыток* — определяет, сколько раз система должна пытаться вы полнить операцию, прежде чем сообщать об ошибке
- T Протокол NFS обычно функционирует на основе UDP , а не TCP-пакетов. Опция -T заставляет систему использовать протокол TCP. Это имеет смысл при подключении к NFS-серверу, расположенному в другом сетевом сегменте так как способствует повышению надежности связи.
- U Эта опция противоположна T она включает режим принудительного ис пользования UDP-пакетов, даже если обе стороны готовы установить TCP соединение
- -i — помечает точку монтирования как *прерываемую* Смысл флага в том что вызывающая программа получает сообщение об ошибке когда операция за вершается по причине тайм-аута
- -s — позволяет создать *нежесткую точку монтирования* Смысл флага в том, что после заданного числа попыток (устанавливается с помощью опции -R) файловая система возвращает вызывающей программе сообщение об ошибке не позволяя ей "зависнуть"

Если экспортируемый каталог не удается смонтировать, то не исключено, что сер вер не сконфигурирован на предоставление FreeBSD-клиенту доступа к каталогу Свяжитесь с администратором сервера и узнайте, в чем дело. Если вы являетесь ад министратором обеих систем, обратитесь к главе 18, в которой рассказывалось о кон фигурировании NFS-сервера во FreeBSD, или к документации, если операционная система сервера — не FreeBSD.

Чтобы сделать NFS каталог постоянно доступным, добавьте запись для него в файл /etc/fstab В ней должны быть заданы имя NFS-сервера и имя экспортируе

мого каталога. Например, в следующей строке каталог /opt/Office сервера nfsserv монтируется к каталогу /opt/OpenOffice.org локального компьютера:

```
nfsserv:/opt/Office  /opt/OpenOffice.org  nfs  r  0  0
```

Каталог делается доступным только для чтения (даже если на сервере он экспортируется с правом чтения/записи). Чтобы подключить его, перезагрузите систему или введите **mount -a**. Из экспортируемого каталога можно запускать программы, читать файлы и т.д., как если бы это был локальный каталог.

Точки совместного доступа SMB/CIFS

SMB (Server Message Block — блок серверных сообщений), известный также как CIFS (Common Internet File System — единая файловая система для Internet), представляет собой протокол совместного доступа к файлам, используемый в Microsoft Windows и OS/2. В главе 18 рассматривался серверный пакет Samba, реализующий поддержку этих протоколов во FreeBSD. Samba позволяет FreeBSD-системе играть роль сервера для клиентов Windows. Но иногда требуется противоположное: получить доступ к совместным каталогам Windows или Samba из FreeBSD. Для этого существуют две утилиты, smbclient и mount_smbfs. Первая из них является частью пакета Samba и по своему интерфейсу напоминает утилиту ftp. Вторая входит в состав FreeBSD и позволяет монтировать каталоги SMB/CIFS, как это делает утилита mount_nfs в отношении каталогов NFS.

Утилита smbclient

В первом приближении утилиту smbclient можно считать аналогом ftp. Их функционирование и поддерживаемые команды сходны. Рассмотрим пример:

```
$ smbclient //smbserver/sjones
added interface ip=192.168.1.6 bcast=192.168.1.255 nmask=255.255.255.0
Password:
Domain=[RINGWORLD] OS=[Unix] Server=[Samba 2.2.1a]
smb: \> get license.txt
getting file license.txt of size 356 as license.txt (38.6 kb/s) (
(average 38.6 kb/s)
smb: \> quit
```

Аргументом утилиты является спецификация сервера (//smbserver/sjones), которая состоит из двух частей: имя сервера (smbserver) и имя каталога совместного доступа (sjones). Последнее часто совпадает с именем пользователя, получающего доступ к серверу Samba. Если пароль (не отображается на экране) введен правильно, становятся доступными стандартные ftp-команды, такие как cd, lcd, get, put, dir и help.

После спецификации сервера могут задаваться дополнительные параметры, влияющие на работу утилиты smbclient. Рассмотрим наиболее важные из них.

- *пароль*. Пароль можно указывать в командной строке, но так делать не рекомендуется, поскольку он отображается на экране, сохраняется в списке введенных команд и становится известен команде ps. Единственное исключение — сценарий, осуществляющий автоматическую пересылку файлов.
- **-s** *конфигурационный_файл*. Утилита smbclient читает конфигурационные параметры из того же файла smb.conf, который содержит установки сервера Samba. Во FreeBSD он обычно находится в каталоге /usr/local/etc, но можно подключить и другой файл.
- **-U** *пользователь*. По умолчанию утилита smbclient использует имя локального пользователя и при доступе к удаленной системе. Эту установку можно изменить, указав имя нужного пользователя с помощью опции -U.

- `-I` *IP-адрес* По умолчанию утилита `smbclient` подключается к компьютеру, имя NetBIOS которого указано в качестве имени службы Механизм распознавания имен не всегда работает правильно, поэтому можно заставить утилиту проигнорировать имя NetBIOS и подключиться к компьютеру с заданным IP адресом
- `W` *рабочая_группа*. В NetBIOS компьютеры разбиваются на *рабочие группы* или *домены*. В файле `smb.conf` задана рабочая группа, соответствующая локальной системе, но иногда нужно переопределить эту установку, чтобы подключиться к компьютеру из другой рабочей группы, сконфигурированному на прием запросов только от своей локальной группы Опция `W` заявляет иную групповую принадлежность локального компьютера

В целом утилита `smbclient` — удобное средство, если нужно быстро скопировать несколько файлов с одного компьютера на другой. Поскольку в Windows-системах часто работают серверы SMB/CIFS, обеспечивающие поддержку одноранговых сетей, утилита `smbclient` позволяет им обмениваться файлами, как если бы это были FTP-серверы. Пакет Samba и утилиту `smbclient` можно даже использовать вместо FTP между FreeBSD и другими UNIX-системами, поскольку в протоколах SMB/CIFS поддерживается шифрование паролей Но в то же время утилиту `smbclient` нельзя считать полноценным клиентом совместного доступа к файлам Если необходимо, чтобы программа непосредственно выполняла чтение и запись файлов на сервере воспользуйтесь другой утилитой `mount_smbfs`

Команда mount_smbfs

Команда `mount_smbfs` позволяет работать с каталогом SMB/CIFS так, как если бы это был локальный раздел или съемный диск В упрощенном виде синтаксис команды выглядит так:

```
mount_smbfs [-I узел] [-N] [-W рабочая_группа] [-f режим] [-d режим]\
[-u UID] [-g GID] //пользователь@сервер/каталог точка_монтирования
```

Назначение опций таково

- `-I` *узел* — эквивалент опции `-I` утилиты `smbclient` но здесь в качестве аргумента указывается IP-адрес или имя компьютера
- `-N` По умолчанию команда `mount_smbfs` запрашивает пароль С помощью опции `-N` можно заставить ее искать пароль в пользовательском файле `/.nsmbrc` (рассматривается ниже).
- `-W` *рабочая_группа* — задает рабочую группу как и аналогичная опция утилиты `smbclient`
- `-f` *режим* Протоколы SMB/CIFS не поддерживают характерные для UNIX механизмы прав доступа. Опция `-f` сообщает FreeBSD о том, какие права доступа нужно назначать файлам Например, опция `-f 0644` включает режим доступа 0644 (`-rw-r--r--`) для всех файлов совместно используемого каталога
- `-d` *режим* Эта опция аналогична `-f` но применяется к каталогам, а не к файлам
- `-u` *UID* — задает идентификатор пользователя, назначаемый всем файлам каталога SMB/CIFS. Например, опция `-u 1002` говорит о том, что владельцем файлов станет пользователь с идентификатором 1002 По умолчанию выбирается идентификатор, соответствующий владельцу точки монтирования
- `-g` *GID* Эта опция аналогична `-u`, но задает идентификатор группы, назначаемый всем файлам каталога SMB/CIFS

- *//пользователь@сервер/каталог* — обязательная опция, которая задает имя пользователя для совместного доступа к каталогу, имя сервера в NetBIOS и имя каталога на сервере
- *точка_монтирования* — локальная точка монтирования, предназначенная для хранения файлов совместно используемого каталога

В качестве примера рассмотрим следующую команду:

```
# mount_smbfs -u 65534 -f 0444 -d 0555 //sjones@smbserver/images%
/usr/local/graphics
```

Эта команда монтирует каталог `images` компьютера `smbserver` к каталогу `/usr/local/graphics`, используя учетную запись `sjones`. Файлы будут принадлежать пользователю `nobody` (идентификатор 65534) и будут доступны только для чтения всем пользователям (режим доступа 0444 для файлов и 0555 для каталогов).

Команда `mount_smbfs` выдаст запрос на ввод пароля, если отсутствует опция `-N`. В противном случае необходимо наличие файла `/usr/local/etc/nsmb.conf` или файла `~/.nsmbrc` в начальном каталоге пользователя, выполняющего команду (обычно это `root`). Информация в этом файле разбита на группы, соответствующие серверам. Информационный блок, необходимый для монтирования каталога без ввода пароля, выглядит так:

```
[сервер:пользователь.каталог]
password=пароль
```

При желании можно добавить другие параметры. Например, строка `workgroup=`*имя_группы* задает рабочую группу, что позволяет не указывать опцию `-W` в командной строке. С помощью строки `addr=`*IP адрес* можно задать IP-адрес сервера.

Если пароли хранятся в файле `nsmb.conf` или `.nsmbrc`, то они могут стать доступными другим пользователям. Назначьте файлу режим 0600, чтобы никто, кроме владельца, не мог читать его. Файл `nsmb.conf` должен принадлежать пользователю `root`, а файл `.nsmbrc` — пользователю, выполняющему команду `mount_smbfs`.

Обычно команда `mount_smbfs` доступна только суперпользователю. Если необходимо дать рядовым пользователям возможность монтировать каталоги SMB/CIFS, установите для исполняемого файла `mount_smbfs` бит SUID, как описывалось в главе 8.

```
# chmod a+s /sbin/mount_smbfs
```


Это изменение чревато неприятностями, если в исполняемом коде команды содержится ошибка, злоумышленник может использовать ее непредсказуемым образом. Но даже в отсутствие ошибок небезопасна сама возможность монтирования каталогов SMB/CIFS. К примеру, злоумышленник может незаметно для остальных пользователей смонтировать каталог поверх такого стандартного каталога FreeBSD, как `/etc`, изменив его структуру.

По окончании работы с каталогом, который смонтирован командой `mount_smbfs`, его нужно демонтировать. Это делается традиционным способом с помощью команды `umount`:

```
# umount /usr/local/graphics
```

Клиенты дистанционной регистрации

Клиенты дистанционной регистрации позволяют работать с удаленными системами как с локальными. Можно запускать программы в дистанционном режиме, просматривать файлы, хранящиеся на удаленном компьютере, и т.д. В главе 21 описывались серверы, реализующие такие функции в удаленных системах. В этом же разделе будут представлены два распространенных текстовых клиента

На заметку Графические клиенты дистанционной регистрации рассматривались вместе с соответствующими серверами в главе 21

Клиенты Telnet

Во FreeBSD клиент Telnet довольно прост. Достаточно ввести **telnet** и имя либо IP адрес компьютера, к которому следует подключиться. Если все пройдет успешно, появится приглашение на ввод пароля, после чего пользователь будет зарегистрирован в удаленной системе.

```
$ telnet barsoom.threeroomco.com
Trying 192.168.1.1
Connected to barsoom.threeroomco.com
Escape character is '^]'
Password:
Copyright (c) 1992-2002 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
    The Regents of the University of California. All rights reserved

FreeBSD 5.0-DP1 (GENERIC) #0: Sun Apr  7 02:51:42 GMT 2002
bash-2.05$
```

В этом примере продемонстрирована попытка подключения к Telnet серверу **barsoom.threeroomco.com**. По умолчанию утилита `telnet` во FreeBSD самостоятельно высылает имя текущего пользователя серверу, поэтому его вводить не нужно, только пароль (не отображается на экране). Если же необходимо зарегистрироваться в системе от имени другого пользователя, задайте в командной строке опцию `-l` *имя_пользователя*.

После регистрации в системе можно запускать любые программы текстового режима: редакторы, почтовые клиенты, компиляторы и т.д. Можно даже выполнить команду `su`, чтобы получить полномочия суперпользователя и доступ к административной утилите `sysinstall`. Кроме того, в главе 21 описывалась процедура запуска X-приложений из сеанса Telnet.

Внимание Нежелательно вызывать команду `su` через Telnet-соединение. В протоколе Telnet пароли не шифруются, поэтому злоумышленник, воспользовавшись сетевым анализатором, сможет легко узнать пароль суперпользователя. Системным администрированием лучше заниматься по протоколу SSH.

Поскольку в протоколе Telnet все данные, включая пароли, пересылаются в незашифрованном виде, его лучше применять в небольшой изолированной сети с небольшим количеством пользователей. В крупных сетях, имеющих к тому же выход в Internet, удаленный доступ должен осуществляться средствами SSH.

Клиенты SSH

Протокол SSH служит тем же целям, что и Telnet, однако реализует шифрование. Оно применяется не только к именам пользователей и паролям, но и ко всем данным, передаваемым по сети. Подобно утилите `telnet`, клиент `ssh` высылает имя пользова

теля автоматически, поэтому его не нужно указывать при установлении соединения Сеанс регистрации по протоколу SSH выглядит так:

```
$ ssh barsoom.threeroomco.com
The authenticity of host 'barsoom  hreeroomco com (192 168 1 6)
can't be established.
RSA1 key fingerprint is 48:17:8e:1c da:ab:81:fe:01:76:bc:a4:f1:90
f5:00.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'barsoom.threeroomco.com' (RSA1) to the
 list of known hosts.
jcarter@barsoom.threeroomco.com's password:
Last login: Mon Jun  3 00:26.45 2002 from speaker
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
    The Regents of the University of California. All rights reserved
FreeBSD 5.0-DP1 (GENERIC) #0: Sun Apr  7 02:51:42 GMT 2002
bash 2 05$
```

Утилита ssh непривычно "многословна", поскольку это первое соединение с сервером. Криптографические идентификаторы всех серверов, с которыми устанавливались соединения, хранятся в файлах .ssh/known_hosts и .ssh/known_hosts2 в начальном каталоге пользователя. Если информация о сервере не может быть найдена или не соответствует содержимому файлов, утилита ssh сообщает об этом и запрашивает подтверждение на продолжение соединения Это способ защиты от атак, заключающихся в том, что злоумышленник "прикрывается" IP-адресом известного сервера SSH, пытаясь перехватить пароли Если пользователь разрешает продолжать соединение или подлинность сервера устанавливается автоматически, утилита ssh запрашивает пароль, после чего пользователь регистрируется в системе.

С этого момента можно делать то же самое, что и в сеансе Telnet· запускать текстовые программы, переадресовывать вывод X-приложений на локальный дисплей и т д. Как описывалось в главе 21, в SSH поддерживается *туннелирование* трафика других протоколов. Это означает, что подсистема SSH может перехватывать данные, адресуемые портам других протоколов, и пересылать их через SSH-соединение, обеспечивая шифрование незашищенных данных Чаще всего применяется туннелирование X-соединений Чтобы активизировать этот режим, нужно вызвать утилиту ssh с опцией -X. Другой вариант – добавить показанную ниже строку в файл /etc/ssh/ssh_config.

```
ForwardX11 yes
```

Естественно, сервер тоже нужно сконфигурировать на переадресацию X соединений (см главу 21). В файле /etc/ssh/ssh_config можно задавать множество других параметров но в стандартной конфигурации FreeBSD всем им присвоены стандартные значения

Резюме

Сетевые клиенты играют чрезвычайно важную роль во многих системах Одни из них, в частности программы чтения почты и Web-броузеры, позволяют устанавливать связь с внешним миром Другие, такие как клиенты совместного доступа к файлам, используются преимущественно в локальных сетях, способствуя повышению эффективности пользовательской работы Третьи, например FTP-клиенты и клиенты удаленного доступа, могут применяться как в локальных, так и в глобальных сетях, в зависимости от потребностей пользователей В распоряжении пользователей FreeBSD есть множестао клиентских программ каждой категории.

Полный справочник по

Глава 25

Офисные средства

Многие рабочие станции используются для решения лишь нескольких основных задач: обработки текста, вычислений с помощью электронных таблиц, подготовки презентаций, построения диаграмм и т.п. Эти задачи столь важны, что соответствующие программы часто объединяются в единый *офисный пакет*. В среде Windows наиболее широко распространен пакет Microsoft Office, но он недоступен во FreeBSD. В этой главе представлены офисные пакеты FreeBSD, описываются их ключевые особенности, а в качестве примера подробно рассматривается один из наиболее мощных пакетов: OpenOffice.org. Естественно, офисные пакеты слишком сложны, поэтому приводимая информация по большому счету носит вводный характер.

Офисные пакеты FreeBSD

Во FreeBSD существует довольно много офисных систем. Наиболее популярные из них — GNOME Office, KOffice и OpenOffice.org.

GNOME Office

Пакет GNOME Office не замышлялся как единый унифицированный набор программ. Его разработчики просто объединили независимые программы. Впрочем, компоненты пакета используют общие библиотеки функций, в частности GTK+ (GIMP Tool Kit), которая была разработана для программы GIMP, а впоследствии принята GNOME и другими программами. Ведутся работы по обеспечению совместного доступа к данным и унификации форматов файлов, используемых офисными программами. Но в то же время нельзя сказать, что все эти программы составляют некий изолированный набор и они должны использоваться совместно. Можно инсталлировать любой из компонентов отдельно и осуществлять обмен файлами между ним и какой-нибудь другой, не связанной с пакетом программой.

Полный список компонентов GNOME Office доступен по адресу `http://www.gnome.org/gnome-office`. В этот список входят компоненты четырнадцати категорий: редактор электронных таблиц, текстовый процессор, коммуникационные программы, Web-броузер, программы векторной и растровой графики, программы просмотра изображений, почтовые клиенты, средства автоматизации, коллективной работы, средства построения графиков и диаграмм, программы управления проектами, средства финансового планирования, программы подготовки презентаций и СУБД. Наиболее значительные компоненты пакета перечислены ниже.

- **Gnumeric** — программа табличных вычислений. По заявлениям разработчиков она реализует 95% функций Microsoft Excel. Программный файл называется `gnumeric`.
- **AbiWord** — текстовый процессор. Разрабатывавшийся как кросс-платформенное средство AbiWord доступен в различных UNIX-системах, а также в Windows и BeOS. Программный файл называется `AbiWord` (с символической ссылкой `abiword`).
- **Galeon**. Web-броузер Galeon, упоминавшийся в главе 24, считается частью пакета GNOME Office. Программный файл называется `galeon`.
- **GIMP**. Программа GIMP (GNU Image Manipulation Program) – наиболее мощный редактор растровой графики во FreeBSD, входящий в состав GNOME Office. Подробнее о GIMP рассказывается в главе 26. Программный файл называется `gimp`.

- **Dia** — программа создания диаграмм. Ее аналогом в Windows является Visio, хотя Dia менее сложна. Программный файл называется `dia`.
- **Guppi** — программа построения графиков. Она способна принимать данные, сгенерированные другими программами, и потому является весьма ценным инструментом финансовых и научных исследований. Программный файл называется `guppi-numeric`.
- **Balsa** — почтовый клиент, аналогичный программам, описанным в главе 24. Программный файл называется `balsa`.
- **Evolution** — еще один почтовый клиент пакета GNOME Office, с более широкими возможностями, чем Balsa. Программный файл называется `evolution`.
- **Gnucash**. Пользователи, работающие дома, часто нуждаются в персональных средствах финансового планирования, а Gnucash — одна из наиболее популярных программ такого рода во FreeBSD. Программный файл называется `gnucash`.
- **GNOME-DB** — это СУБД пакета GNOME Office. Программный файл называется `gnomedb-fe`.
- **OpenOffice.org**. На Web-узле пакета GNOME Office имеется предупреждение о том, OpenOffice.org не является полностью интегрированным компонентом. В настоящее время OpenOffice.org считается конкурирующим пакетом, и именно в таком виде он будет представлен далее в главе. Основной программный файл называется `soffice`, но другие подкомпоненты могут вызываться иначе.

В отличие от интегрированных офисных пакетов компоненты GNOME Office инсталлируются независимо. Большинство из них доступно в разделе `gnome` коллекции пакетов или коллекции переносимых приложений. В качестве исключений отметим AbiWord (раздел `editors`), Sketch (раздел `graphics`) и Guppi (раздел `math`).

К числу достоинств GNOME Office следует отнести наличие программ GIMP (сопоставима по функциональным возможностям с Adobe Photoshop), Gnucash (приближается по возможностям к Quicken), Gnumeric (популярный редактор электронных таблиц) и OpenOffice.org. Последний пакет является наиболее близким аналогом Microsoft Office во FreeBSD. Остальные программы, например AbiWord, несмотря на то что они вполне функциональны, лишены ряда свойств, к которым привыкли пользователи Windows. Конечно, это не всегда является недостатком: ведь философия UNIX традиционно заключается в том, чтобы много маленьких программ можно было объединять, добиваясь функциональных возможностей, характерных для более крупных приложений Windows. Кроме того, интегрированные офисные пакеты нередко критикуют за чрезмерные требования к ресурсам системы.

KOffice

Пакет KOffice играет в оболочке KDE (K Desktop Environment) такую же роль, как и GNOME Office в GNOME. Однако он изначально разрабатывался как офисный пакет KDE, в отличие от GNOME Office, который появился в результате объединения множества независимых проектов. Дополнительную информацию о KOffice можно найти на Web-узле `www.koffice.org`.

Пакет KOffice содержит меньшее число компонентов и категорий компонентов, чем GNOME Office, хотя наиболее важные типы представлены в обоих пакетах. Пакет KOffice составляют следующие программы:

- **KWord** — текстовый процессор, более функциональный, чем AbiWord.
- **KSpread** — редактор электронных таблиц. В нем не реализованы многие функции Microsoft Excel или Gnumeric, но в целом это качественный продукт.

- **KPresenter** — программа подготовки презентаций. Образцы презентаций доступны по адресу `http://www kde.org/kdeslides/`.
- **KVivio** — редактор диаграмм, напоминающий Windows-программу Visio или Dia из GNOME Office. KVivio реализует систему подключаемых сценариев, что расширяет функциональные возможности программы. Например, можно написать сценарий, который принимая файл заголовков C++, будет строить диаграмму классов.
- **Kontour** — редактор векторной графики.
- **Krita** — редактор растровой графики. На Web-узле KOffice он сравнивается с GIMP и Adobe Photoshop, но пока что его функциональные возможности далеко не эквивалентны.
- **Kugar** — эта программа позволяет автоматизировать создание деловых отчетов. Единственный компонент KOffice, не имеющий аналога в GNOME Office.
- **KChart** — еще один графический редактор, позволяющий строить диаграммы и графики на основе файлов данных.

На заметку Тот факт, что у KOffice меньше компонентов, чем у GNOME Office, еще ни о чем не говорит, поскольку у некоторых приложений GNOME Office есть аналоги в KDE, не являющиеся официальной частью KOffice. Например, программы KMail и Konqueror — это почтовый клиент и Web-броузер, но, в отличие от своих эквивалентов в GNOME (Balsa и Galeon), они не входят в состав KOffice.

В целом компоненты KOffice хорошо проявляют себя во многих ситуациях. У них, правда, нет ряда функциональных возможностей, встречающихся у конкурирующих программ. Например, у KWord нет средств сравнения версий и проверки грамматики. Но разработчики KOffice ставят перед собой цель реализовать все средства, к которым привыкли пользователи офисных пакетов Windows, поэтому со временем указанные недостатки будут устранены.

OpenOffice.org

OpenOffice.org (`www.openoffice.org`) — это аналог пакета StarOffice компании Sun (`www.sun.com`), но с открытым исходным кодом. Примерно такое же сходство наблюдается между броузерами Mozilla (открытый исходный код) и Netscape Navigator (коммерческая программа). На момент написания книги пакету StarOffice 6.0 соответствовал пакет OpenOffice.org 1.0. Компания Sun выпускает пакет StarOffice для Windows, Solaris и Linux, но не для FreeBSD, хотя его можно запустить во FreeBSD, используя библиотеки поддержки Linux, как описывалось в главе 4. Что касается OpenOffice.org, то разработчики упорно трудились над тем, чтобы пакет компилировался во FreeBSD, благодаря чему он может свободно работать во FreeBSD версии 4.6 и выше. Правда, иногда FreeBSD-версии пакета демонстрируют определенную неста бильность; в этом случае попробуйте установить Linux-версию.

На заметку Разработчики OpenOffice.org изначально планировали назвать свой продукт *OpenOffice*, однако выяснилось, что это название зарегистрировано и принадлежит другой программе. Поэтому было решено добавить к названию "расширение" *.org*.

Из трех основных офисных пакетов FreeBSD наиболее всего напоминает Microsoft Office именно OpanOffice.org. К тому же в нем содержатся наилучшие фильтры импорта и экспорта файлов Microsoft Office, о чем будет рассказываться на с. 498. В пакет входят следующие компоненты:

- **Writer** — как подсказывает название, это текстовый процессор. Возможно наиболее сложный среди доступных во FreeBSD.
- **Calc** — редактор электронных таблиц, весьма напоминающий Excel, хотя и не являющийся его "клоном".
- **Draw**. Подобно другим офисным пакетам OpenOffice.org содержит графический редактор, только в данном случае доступен лишь редактор векторной графики.
- **Impress** — диспетчер презентации.

До появления версии 6.0 пакет StarOffice включал множество дополнительных компонентов, для запуска которых использовался единый исполняемый файл. Это делало пакет довольно медлительным, особенно при запуске. С появлением StarOffice 6.0 и OpenOffice.org 1.0 число компонентов резко сократилось, что позволило разработчикам сконцентрироваться на базовых функциях пакета. Программы стали запускаться быстрее, чем прежде (хотя все же не так быстро, как во многих других пакетах).

Если необходимо перевести служащих, привыкших работать в Microsoft Office, во FreeBSD, то пакет OpenOffice.org может сыграть важную роль, поскольку затраты на обучение персонала будут минимальными. Компоненты пакета хотя и не идентичны компонентам Microsoft Office, но достаточно похожи на них, так что пользователям почти не придется переучиваться. Доступность пакета в Windows является не менее важным фактором, если планируется организовать работу в смешанной среде.

Вспомогательные средства

Не все офисные средства входят в состав крупных пакетов. Просмотрите списки пакетов в программе `sysinstall` или в каталоге `/usr/ports`, чтобы узнать, какие еще программы имеются в наличии. Укажем наиболее важные из них.

- **LaTeX** — настольная издательская система, которая, в отличие от большинства текстовых процессоров, не основана на модели WYSIWYG (What You See Is What You Get — что видишь на экране, то и получишь при печати). Чтобы применить LaTeX, нужно загрузить файл в текстовый редактор и вставить в него команды форматирования, напоминающие команды HTML, после чего "прогнать" файл через программу LaTeX, которая преобразует его в заданный выходной формат, например PostScript. Программа часто применяется для подготовки научных публикаций, поскольку она великолепно справляется с обработкой формул.
- **LyX** (`www.lyx.org`) — графическая надстройка к LaTeX. LyX содержит набор меню, напоминающих меню текстовых процессоров, хотя и не является WYSIWYG-программой. Тем не менее она будет удобна пользователям, не знакомым с LaTeX.

На заметку Модифицированная версия LyX — KLyX — использует набор элементов управления Qt, на котором основана оболочка KDE. Но разработка KLyX прекращена, поэтому LyX является более предпочтительным средством.

- **abs** — редактор электронных таблиц, предназначенный для X-среды и не зависящий от какого-либо офисного пакета. Он доступен по адресу `http://www.ping.be/bertin/abs.shtml`.
- **Oleo** — официальный GNU-редактор электронных таблиц, работающий как в текстовом, так и в графическом режиме. Он проще, чем Gnumeric.

Поиск программ нужного типа можно выполнять на Web-узле sourceforge.net. Под контролем компании Sourceforge находится множество программ с открытым исходным кодом

Допускается совместное использование компонентов различных пакетов. Кому-то к примеру, больше нравится программа KWord из KOffice, чем AbiWord из GNOME Office, но табличные вычисления ему удобнее делать в Gnumeric, а не в KSpread. Проблема может заключаться в том, что в разных проектах применяются разные инструментальные средства разработки, поэтому программы внешне различаются и при совместной работе потребляют больше памяти. Кроме того, форматы файлов могут быть несовместимы, и для обмена данными между программами придется прибегнуть к стороннему формату. Также большинство офисных пакетов инсталлируется по принципу "все или ничего", что приведет к излишним затратам дискового пространства. Исключение составляет пакет GNOME Office, компоненты которого инсталлируются независимо.

Особенности офисных пакетов

Пользователям офисных пакетов Windows некоторые особенности аналогичных пакетов во FreeBSD могут показаться непривычными. В этом разделе мы поговорим о том, с какими проблемами чаще всего сталкиваются пользователи и как их обойти.

Обработка шрифтов

Одна из самых больших проблем при обработке текста во FreeBSD заключается в том, что модель шрифтов в X Window System слишком примитивна по современным графическим стандартам. Кроме того, подсистема шрифтов не слишком хорошо интегрируется с подсистемой печати, что еще больше все усложняет. Предположим, в системе инсталлируется некий шрифт, как описывалось в главе 13. Текстовый процессор сможет выводить данные на экран, используя этот шрифт, но программа должна иметь возможность направлять текст на печать в том виде, в каком он представляется на экране. Однако в X-среде шрифты преобразуются в растровую форму, а подсистема печати FreeBSD не сообщает программе разрешающую способность принтера. Другими словами, программа не "знает", какой размер шрифта запрашивать у X-сервера для печати. Трудности проявляются не только в координации между экранными и печатными шрифтами. Для корректного отображения текста программа-редактор должна управлять межсимвольными интервалами, но X Window System не предоставляет программе необходимые для этого данные.

С учетом всего вышесказанного офисные пакеты и текстовые процессоры как правило, содержат собственные уникальные средства обработки шрифтов. Рассмотрим некоторые из них:

- **Визуализация шрифтов в текстовых процессорах.** Некоторые программы, например OpenOffice.org, реализуют механизмы визуализации шрифтов, не зависящие от X Window System. Это означает, что приходится инсталлировать шрифты в текстовом процессоре независимо от системных шрифтов. Такой подход позволяет текстовому процессору непосредственно подготавливать шрифт к печати и в то же время требует наличия у программы собственного кода отображения текста на экране.
- **Установление соответствия между экранными и печатными шрифтами.** Текстовый процессор может поручать X-серверу отображение шрифтов на экране, но закреплять эти шрифты за стандартными шрифтами PostScript-принтера или за

файлами шрифтов инсталлированными независимо в самом текстовом процессоре AbiWord — одна из программ, в которой применяется такой подход

- **Разделение экранных и печатных шрифтов**. Программы LaTeX и LyX игнорируют проблему несоответствия экранных и печатных шрифтов. Для них не предусмотрена совместимость с моделью WYSIWYG, т.е. не ставится задача отображать шрифты на экране так, как они будут выглядеть при печати. Но в такой программе все равно приходится инсталлировать шрифты.
- **Использование расширенного сервера шрифтов**. Коммерческая программа FontTastic реализует все функции традиционного сервера шрифтов (см. главу 22), но также содержит средства, позволяющие интегрировать механизмы обработки экранных и печатных шрифтов. В текстовых процессорах FreeBSD такой подход не реализован, в отличие от нескольких коммерческих пакетов для Linux (ApplixWare и WordPerfect Office 2000).

Печать

Большинство офисных программ FreeBSD ориентировано на то, что принтер уже сконфигурирован (см. главу 9). Эти программы считывают файл `/etc/printcap`, чтобы определить доступные принтеры, и отображают диалоговое окно настроек печати, позволяющее выбрать используемый принтер. Примерно такие же окна выдаются в Windows или Mac OS, поэтому у пользователей, знакомых с ними, не должно возникнуть проблем. Но для некоторых приложений все же требуется вводить имя принтера вручную. К примеру, рассмотрим окно настроек печати программы AbiWord (рис. 25.1). Если пользователь не укажет имя принтера, задание будет направлено на стандартный принтер. Чтобы этого избежать, добавьте `-P`*имя_принтера* к команде печати, где аргумент *имя_принтера* — это название очереди печати из файла `/etc/printcap`.

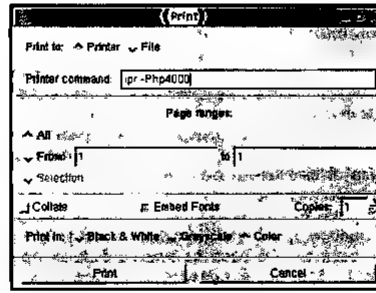


Рис. 25.1. Для некоторых программ имя нужного принтера требуется указывать вручную

Некоторые программы поддерживают дополнительные устройства печати. К примеру, широко используется возможность направления задания в файл на диске. На рис. 25.1 это можно осуществить, выбрав в группе Print to опцию File, а не Printer. В результате обычно получается PostScript-файл, который можно вручную направить на печать (на PostScript-принтер), просмотреть с помощью специальной программы такой как Ghostview, или переслать в другую систему для обработки. Многие офисные

программы в настоящее время умеют создавать файлы PDF (Portable Document Format — формат переносимых документов). Некоторые программы могут выводить данные для факса, что удобно при наличии факс-модема. Если программа не реализует такой режим, инсталлируйте соответствующую утилиту например Gfax из пакета GNOME Office. Эта утилита формирует очередь на печать обработчик которой определяет номер факса, а затем посылает документ в факсовом формате

Большинство программ FreeBSD предполагает, что принтер является PostScript совместимым Как описывалось в главе 9, можно заставить систему преобразовывать PostScript-задания на печать, генерируемые приложениями, в формат, который будет понятен большинству моделей принтеров, так что отсутствие PostScript-принтера не является проблемой. Но есть и программы, которые могут выводить данные в других форматах. При использовании собственных драйверов принтера такие программы способны выдавать более качественные результаты Однако в этом случае придется создать "неструктурированную" очередь на печать чтобы обработчик не пытался интерпретировать выходные данные как PostScript-данные, что может привести к их повреждению

Импорт и экспорт файлов

Поскольку наиболее популярный офисный пакет в мире — Microsoft Office — не доступен во FreeBSD, для офисных пакетов FreeBSD чрезвычайно важна возможность импорта и экспорта файлов в формате Microsoft Office Это не имеет значения при работе во FreeBSD, но зато позволяет обмениваться документами с пользователями других систем Определенную роль могут играть и средства поддержки других текстовых форматов

Как правило, файлы текстовых процессоров сложнее всего импортировать и экспортировать Несмотря на то что текстовые процессоры основных офисных пакетов FreeBSD способны обрабатывать файлы Microsoft Word, лишь в OpenOffice.org 1.0 (и, соответственно, в StarOffice 6.0) распознаются более-менее сложные документы. Например, AbiWord корректно загружает текст, но при этом теряется большая часть информации о форматировании Иногда в процессе импорта файла программа вообще дает сбой. KWord справляется с задачей гораздо лучше, но не распознает некоторые символы, например маркеры списков, и не извлекает внедренные графические файлы Лишь о программе Writer пакета OpenOffice.org можно сказать, что она воспроизводит почти все аспекты форматирования файла, включая шрифты маркеры списков, исправления и рисунки, хотя и она не идеальна. В частности, при регулярном обмене файлами с Microsoft Word ошибки форматирования имеют тенденцию накапливаться и проявляться, например, в смещении отступов в списках Что касается импорта и экспорта документов в форматах таких текстовых процессоров, как Corel WordPerfect и Lotus WordPro, то они реализованы неидеально в любом офисном пакете FreeBSD

В целом, текстовые процессоры FreeBSD нельзя назвать идеальным средством импорта/экспорта файлов, если только речь не идет о файлах программ, которые сами могут работать во FreeBSD. Можно, например, запустить Linux-версию WordPerfect 8 или даже Microsoft Word под управлением WINE как описывалось в главе 4 Очень хорошие средства импорта/экспорта реализованы в OpenOffice.org, особенно в отношении документов Microsoft Word. Аналогичные средства AbiWord и KWord лучше применять к простым документам или для восстановления текста из файлов

Совет

Если заранее известно о том, что с документом придется работать в другой программе, попробуйте сохранить его в нескольких форматах. Например, документ Microsoft Word можно сохранить в "родном" формате, формате RTF и в виде простого текста. Тогда, если один из механизмов импорта даст сбой, можно будет воспользоваться другим

Импорт и экспорт электронных таблиц осуществляются качественнее, чем в случае текстовых документов. Программы Gnumeric и KSpread способны импортировать и экспортировать большинство электронных таблиц Microsoft Excel. Потенциальная сложность возникает, если импортирующая программа не поддерживает ту или иную функцию, используемую в заданном файле. В таком случае результат может оказаться непредсказуемым.

Форматы растровой графики довольно четко стандартизированы. Большинство графических редакторов поддерживает такие форматы, как TIFF (Tagged Image File Format — теговый формат графических файлов), JPEG (Joint Photographic Experts Group — формат, разработанный группой экспертов в области фотографии) и PNG (Portable Network Graphics — формат переносимой сетевой графики). Загрузка и сохранение таких файлов не вызывают проблем. Форматы векторной графики редко бывают столь же переносимыми. Один из наиболее распространенных форматов обмена файлами между векторными редакторами — EPS (Encapsulated PostScript – инкапсулированные PostScript данные). Но большинству программ все же требуются собственные форматы для полной реализации заложенных в них возможностей.

Образец офисного пакета: OpenOffice.org

В качестве примера офисного пакета в этом разделе будет рассматриваться пакет OpenOffice.org. Многие операции должны быть знакомы пользователям офисных продуктов других платформ, хотя, конечно, есть нюансы, специфичные для FreeBSD.

На заметку

На тот момент, когда писалась эта книга, пакет OpenOffice.org лучше всего работал в Linux-варианте, для которого требуются библиотеки поддержки Linux. Возможно, к тому моменту, когда книга попадет в руки читателей, появятся стабильные FreeBSD-версии пакета.

Конфигурирование шрифтов и принтеров

В пакет OpenOffice.org входит утилита `spadmin`, которая находится в подкаталоге `program` инсталляционного каталога и предназначена для конфигурирования шрифтов и принтеров. Необходимость в ней возникает, когда имя нужного принтера не появляется в окне настроек печати или требуется добавить новый шрифт. Для запуска утилиты достаточно ввести ее имя (с указанием пути, если путь к подкаталогу `program` не задан в переменной среды `PATH`). Окно утилиты представлено на рис. 25.2.

На заметку

При первом запуске утилиты `spadmin` список принтеров скорее всего окажется неполным. Но как только будет вызвана любая программа пакета, она обнаружит принтеры, перечисленные в файле `/etc/printcap`, и добавит их в список доступных принтеров. С помощью утилиты `spadmin` можно также сообщить остальным программам о том, что тот или иной принтер не соответствует стандарту PostScript.

Добавление принтеров

Если требуется сконфигурировать принтер, который не обнаруживается автоматически или не является стандартным PostScript-принтером, выполните перечисленные ниже действия.

1. Щелкните на кнопке New Printer в окне Printer Administration (рис. 25.2). Появится серия диалоговых окон, которые упрощают процесс настройки. В первом окне пользователь должен указать тип добавляемого устройства: обычный принтер, факс или преобразователь PDF.

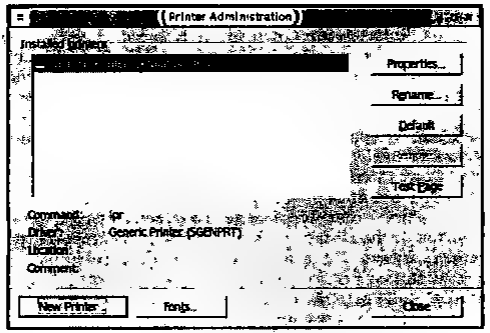
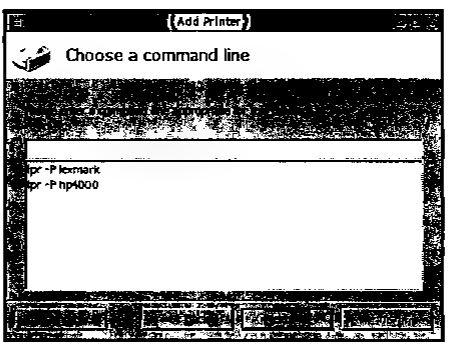


Рис 25.2. В пакете OpenOffice.org, как и в других офисных системах, имеется собственная конфигурационная утилита

2. Выберите переключатель Add a Printer и щелкните на кнопке Next Будет выдан список принтеров большинство которых являются PostScript-моделями
3. Выберите нужный принтер или совместимую модель и щелкните на кнопке Next Появится окно со списком команд, которые можно использовать для печати на принтере как показано ниже.



4. Если одна из команд подходит, выберите ее В противном случае введите команду в поле над списком. По завершении щелкните на кнопке Next
5. Далее утилита запросит имя принтера. Введите его и щелкните на кнопке Finish Теперь имя принтера должно появиться в списке Installed printers (рис. 25.2)

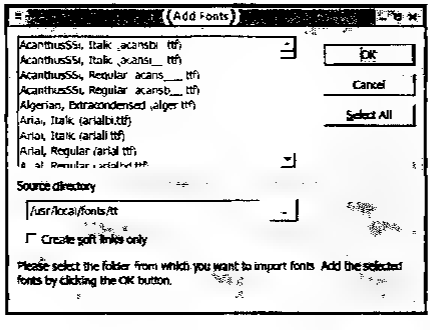
С этого момента принтер должен стать доступным программам пакета OpenOffice org. Иногда чтобы новое определение вступило в силу нужно перезапустить программу

Добавление шрифтов

Во FreeBSD-программах принципы обработки шрифтов зачастую существенно отличаются. В OpenOffice.org могут использоваться X-шрифты, но пакет гораздо лучше работает со шрифтами TrueType или Type 1, которые он обрабатывает самостоятельно. В таких шрифтах распознается гораздо большее число символов, например длинное тире или различные виды кавычек. Поддерживается также сглаживание шрифтов, благодаря чему повышается разборчивость текста. Но эта возможность зависит от наличия пакета XFree86 версии 4.x. В более ранних версиях шрифты OpenOffice.org не сглаживаются.

Для добавления шрифтов к пакету используется уже знакомая нам утилита `spadmin`. Порядок действий при этом таков.

1. Поместите файлы шрифтов в нужный каталог FreeBSD или смонтируйте каталог в котором они хранятся.
2. В окне Printer Administration утилиты `spadmin` (рис. 25.2) щелкните на кнопке Fonts. Появится окно со списком инсталлированных шрифтов.
3. В окне списка щелкните на кнопке Add. Появится окно Add Fonts со списком шрифтов, доступных в исходном каталоге, как показано ниже.



4. Щелкните на кнопке справа от поля Source directory и в появившемся окне выберите нужный каталог шрифтов. После небольшой паузы утилита `spadmin` проанализирует названия шрифтов и обновит список доступных шрифтов.
5. Выберите все шрифты, которые собираетесь инсталлировать, и щелкните на кнопке OK. На короткое время появится индикатор выполнения, после чего снова отобразится окно со списком инсталлированных шрифтов. Щелкните на кнопке Close, чтобы закрыть его и вернуться к окну Printer Administration.
6. Если в системе есть работающие компоненты OpenOffice.org, выйдите из программ и перезапустите их, чтобы сделать новые шрифты доступными.

Пакет OpenOffice 1.0 поддерживает шрифты TrueType и Type 1, а это значит, что он сможет работать практически со всеми шрифтами, доступными в Internet и в коллекциях на компакт-дисках. Правда, наблюдается некоторая несогласованность, если один и тот же шрифт инсталлирован в OpenOffice.org и в X Window System. Иногда используется шрифт X-сервера, а иногда - внутренний шрифт. Это усложняет управление шрифтами.

Совет

Можно сделать шрифт доступным в X Window System под именем, отличным от того, что указано в файле шрифта. Для этого измените название шрифта в файле `fonts.dir` в каталоге X-шрифтов. Затем инсталлируйте шрифт в OpenOffice.org под его "родным" именем и никакого конфликта не возникнет

Создание документа

Чтобы создать новый документ OpenOffice.org, запустите соответствующую программу, введя имя исполняемого файла в командной строке, например **swriter** для Writer, **sdraw** для Draw, **scalc** для Calc или **simpress** для Impress. В каталоге исполняемых файлов OpenOffice.org содержится также ряд программ для запуска компонентов пакета в специальных режимах. Скажем, если ввести **sweb**, то программа Writer будет вызвана в режиме HTML редактора, а если ввести **sfax**, то редактор Writer запросит параметры факса, после чего перейдет в режим создания факсовых документов.

Следует также отметить, что в процессе инсталляции пакета ссылки на его программы могут добавляться в настольные меню. В этом случае программы можно вызывать с помощью мыши.

Совет

Если программа OpenOffice.org работает некорректно, запустите ее из терминального окна. Возможно, будут выданы сообщения об ошибках, которые позволят выяснить причину сбоев. При запуске программы из настольного меню консольные сообщения получить нельзя.

Вызванная программа создает пустой документ, с которым сразу же можно работать. Если необходимо создать другой документ, выберите его тип в меню File⇨New. Например, команда File⇨New⇨Spreadsheet создает новый документ Calc. В любой программе OpenOffice.org можно создать документ любого типа.

Чтобы загрузить существующий документ, выполните команду File⇨Open. Появится стандартное окно выбора файла. Перейдите в каталог, где хранится нужный документ, выберите файл документа и щелкните на кнопке Open.

Внимание

Прежде чем импортировать документ, созданный другой программой, сохраните открытый документ. Несмотря на то что пакет OpenOffice.org 1.0 весьма надежно импортирует документы, существует вероятность сбоя программы при обработке поврежденного или слишком сложного документа.

Редактирование текста

Программа Writer мало чем отличается от других текстовых процессоров (рис. 25.3). У нее есть привычная строка меню, панели инструментов, расположенные под строкой меню и по левой стороне окна, линейка, позволяющая задавать позиции табуляции и поля документа, строка состояния в нижней части окна, где отображается счетчик страниц, и, конечно же, основная область редактирования.

Редактирование текста осуществляется так же, как и в любом текстовом процессоре. Позицию вставки текста обозначает *курсор* — мерцающая вертикальная черта. С помощью мыши можно позиционировать курсор и выполнять перемещение блоков текста. Для форматирования текста нужно выделить фрагмент и щелкнуть на соответствующей кнопке панели инструментов, либо выбрать команду меню, либо нажать определенное сочетание клавиш.

По умолчанию для измерения расстояний программа использует метрические единицы. Если необходимо поменять систему измерений, выполните следующие действия:

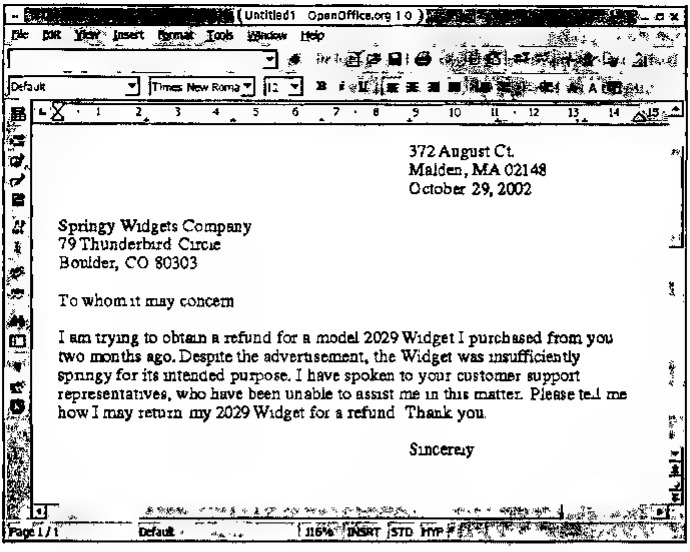
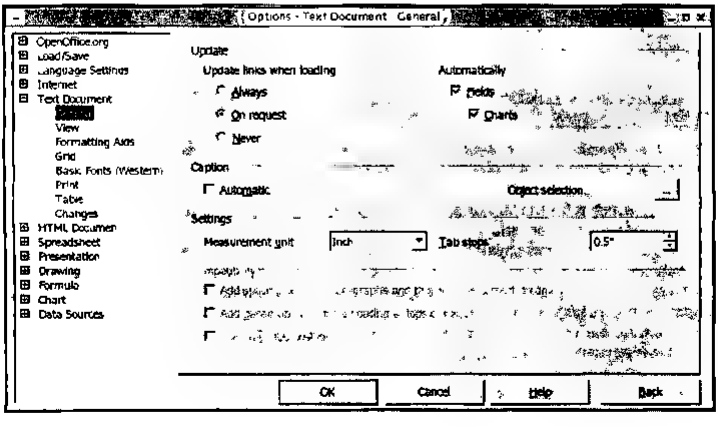


Рис 25.3. Текстовый процессор пакета OpenOffice.org покажется знакомым пользователям других аналогичных редакторов

1 Выберите в меню Tools команду Options. Появится показанное ниже окно Options



2. В левой панели окна перейдите в раздел Text Document и активизируйте пункт General
3. Выберите желаемую единицу измерений в поле Measurement unit.
4. Щелкните на кнопке OK чтобы изменения вступили в силу.

На заметку Если помимо единицы измерений поменять также интервал табуляции то это не отразится на открытых документах. Изменение скажется лишь на новых документах В редактируемом документе позиции табуляции меняются с помощью линейки

В окне Options можно менять не только единицу измерения но и другие установки часть которых относится также к другим компонентам пакета

Табличные вычисления

Как по внешнему виду (рис 25.4), так и по набору функциональных возможностей Calc — типичный редактор электронных таблиц. Пользователи других табличных редакторов не будут иметь проблем с программой Ниже рассмотрены особенности программы

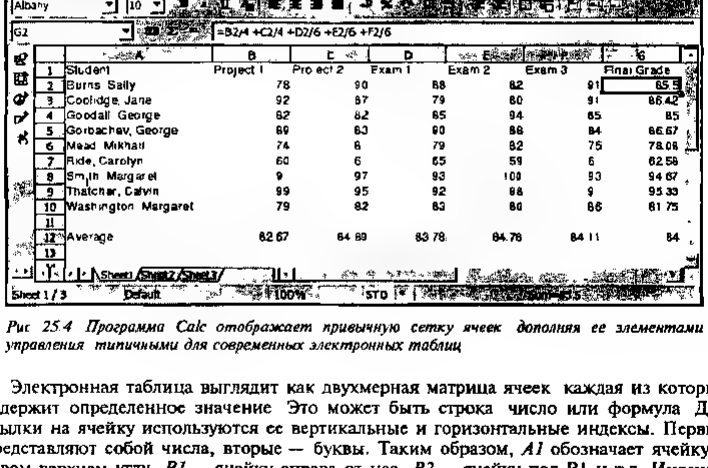


Рис 25.4 Программа Calc отображает привычную сетку ячеек дополняя ее элементами управления типичными для современных электронных таблиц

Электронная таблица выглядит как двухмерная матрица ячеек каждая из которых содержит определенное значение Это может быть строка число или формула Для ссылки на ячейку используются ее вертикальные и горизонтальные индексы. Первые представляют собой числа, вторые — буквы. Таким образом, *A1* обозначает ячейку в левом верхнем углу, *B1* — ячейку справа от нее, *B2* — ячейку под B1 и т.д Индексы отображаются сверху и слева от области данных

Как видно на рис. 25.4, ячейки нередко содержат текст, идентифицирующий данные в конкретных строках или столбцах таблицы На рис. 25.4 такими данными являются оценки студентов. Чтобы ввести данные в ячейку, щелкните на ней мышью и задайте нужное значение в поле над областью данных. По завершении ввода данных

нажмите <Enter> или щелкните на значке с "галочкой" слева от поля (на рис 25 4 этот значок отсутствует, так как он появляется только при активизации поля)

Работа с формулами осуществляется чуть иначе. Формула начинается со знака равенства, который отличает ее от обычной строки. Формулы чаще всего ссылаются на ячейки по именам вида *A4* и могут содержать стандартные арифметические операторы сложения (+), вычитания (-), умножения (*) и деления (/). Допускается использование констант и круглых скобок. Например, формула *=(B2 + C2)/2* вычисляет среднее значение ячеек *B2* и *C2*

Формулы могут включать в себя предопределенные функции Чтобы вставить функцию, щелкните на значке с изображением клавиатуры (третий слева от поля ввода данных) и выберите команду Insert⇨Function или нажмите <F2>. В результате появится окно представленное на рис 25 5. Выполните двойной щелчок на названии функции в списке Function и задайте в поле Number или Formula имя ячейки к которой применяется функция Если название функции известно заранее, ее можно ввести непосредственно в поле ввода данных в основном окне программы. Некоторые функции работают с рядами ячеек. Чтобы задать ряд, укажите имя первой и последней ячеек, разделив их двоеточием. Например, выражение *B2:G2* обозначает все ячейки от B2 до G2 включительно

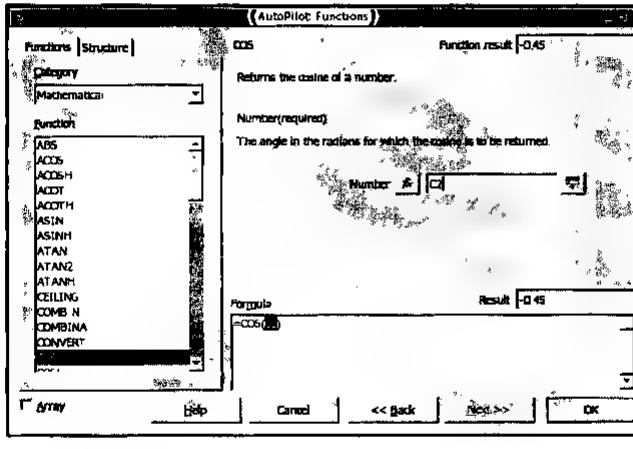


Рис 25 5 Окно выбора функции

Иногда необходимо скопировать созданную формулу, чтобы применить ее к другим ячейкам. Например, на рис. 25.4 одна и та же формула используется для вычисления итоговой оценки каждого студента, хотя ссылки на ячейки в каждом случае разные. Вводить такое количество формул, вручную изменяя необходимые ссылки слишком утомительно Вместо этого нужно задать формулу один раз, после чего выделить ячейку, выбрать команду Edit⇨Copy (или нажать <Ctrl+C>), выделить группу ячеек, к которым применяется формула, и выполнить команду Edit⇨Paste (<Ctrl+V>) Программа Calc скопирует формулу во все ячейки, автоматически скорректировав ссылки Так, если скопировать формулу ячейки G2 на рис. 25 4 в ячейки G3—G10, то каждая копия будет содержать ссылку на соответствующую строку а не на строку 2

Чтобы зафиксировать ссылку на определенную ячейку или ряд ячеек, поставьте перед идентификатором строки или столбца символ '$'. Например, выражение *$B$2* ссылается на ячейку B2 даже если формула, содержащая это выражение, копируется в другую ячейку

Создание векторной графики

Программа Draw предназначена для создания диаграмм, схем, рисунков и прочих графических изображений, не связанных с конкретными данными. На рис. 25.6 изображено окно программы с несложным рисунком

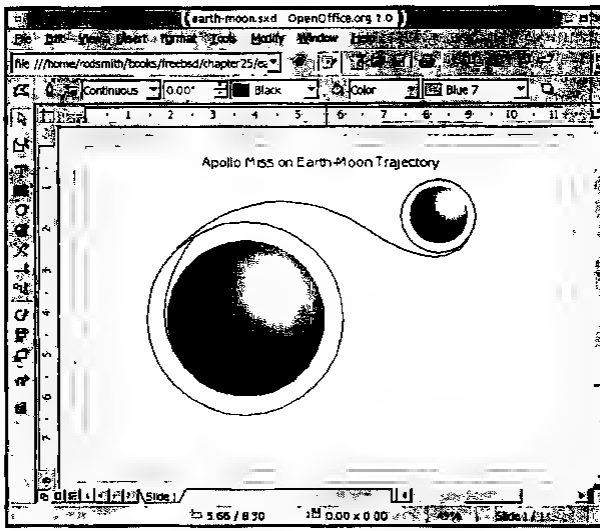


Рис. 25.6. В программе Draw поддерживаются двухмерные и трехмерные графические примитивы

Если необходимо построить диаграмму или график на основе имеющихся данных, сделайте это в программе Calc.

Значки, расположенные в левой части окна программы, предоставляют доступ к различным классам графических инструментов, таких как средства манипулирования текстом, прямоугольники, окружности, линии и т.д. Процедура создания рисунка выглядит так:

1. Наведите указатель мыши на значок, соответствующий выбранному классу инструментов, после чего нажмите и удерживайте левую кнопку мыши. Появится меню с перечнем доступных инструментов. Например, в классе прямоугольных форм на выбор предлагаются фигуры с заливкой и без, со скругленными и прямыми углами

2 Выберите нужную фигуру, например квадрат со скругленными углами и заливкой

3 Задайте дополнительные параметры, такие как цвет и толщина линии. Это делается с помощью соответствующих элементов панелей инструментов или команд меню

4 Убедившись в том, что графическая фигура выбрана (ее значок должен выглядеть "вдавленным"), поместите ее в область рисунка. В зависимости от объекта требуется либо выполнить операцию перетаскивания, либо щелкнуть на значке, а затем — в области рисования. Если выбран текстовый объект, будет создана пустая рамка, в которую нужно ввести текст

5 Повторяйте пп. 1–4, пока все необходимые объекты не будут созданы.

Особую роль играет значок со стрелкой, расположенный в верху левой панели. Если он активизирован, то при щелчке на объекте будут отображены его управляющие точки. С их помощью можно изменять размеры объекта. Когда указатель мыши принимает вид перекрестия с четырьмя стрелками, это означает, что к объекту можно применить операцию перетаскивания. Всевозможные команды управления объектом содержатся в контекстном меню, которое вызывается при щелчке правой кнопкой мыши.

Программа Draw хранит изображения в векторном формате, поэтому они печатаются с той разрешающей способностью, которая поддерживается принтером. Это отличает векторный формат от растрового, не обеспечивающего качественное масштабирование. При печати растрового изображения оно может потерять четкость из-за эффекта масштабирования.

На заметку В документ Draw можно вставить растровое изображение. Оно будет масштабироваться так же, как и в растровом редакторе.

Создание презентации

На первый взгляд программа Impress выглядит так же, как и Draw (рис. 25.7). Однако назначением программы является создание серии *слайдов* (одиночных документов, или страниц), каждый из которых можно напечатать на прозрачной пленке для последующего показа с использованием проекционного экрана. Такая демонстрация может проводиться и на экране монитора. Демонстрация последовательности слайдов называется *презентацией*. Для создания слайдов предусмотрено 20 стандартных форм. Большинство из них предусматривает вывод заголовка в верхней части слайда, под которым расположен один или несколько дополнительных элементов. К таковым относятся различные рамки, электронные таблицы, графические заготовки, текст произвольной формы и типовые объекты. В качестве последних можно импортировать произвольные документы OpenOffice.org, но необходимо предусмотреть, чтобы они поместились на одном слайде.

При вызове программы Impress выводится последовательность диалоговых окон, в которых запрашивается различная информация, а именно: нужно ли создавать новую презентацию или редактировать существующую, каков формат презентации (прозрачные пленки, экран монитора и др.) и какой эффект перехода следует применять при смене слайдов (имеет смысл только для экранных презентаций). Далее появляются основное окно программы (рис. 25.7) и диалоговое окно Modify Slide (рис. 25.8). Последнее предназначено для выбора типа слайда. После щелчка на кнопке OK выбранные структурные элементы появятся в основном окне. Редактирование объектов осуществляется так же, как и в соответствующей программе. Например, если в презентацию вставляется электронная таблица, то ее ячейки редактируются аналогично тому, как это происходит в программе Calc. Разрешается выполнять вырезание и вставку из другого документа, а также импортировать документы.

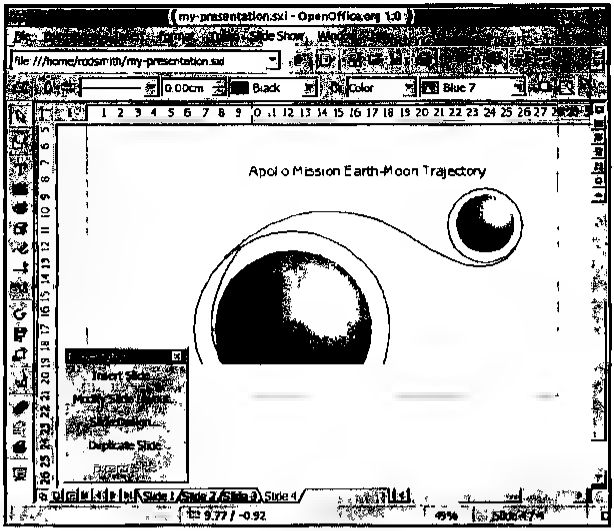


Рис 25 7 Презентация в программе Impress выглядит как последовательность документов Draw

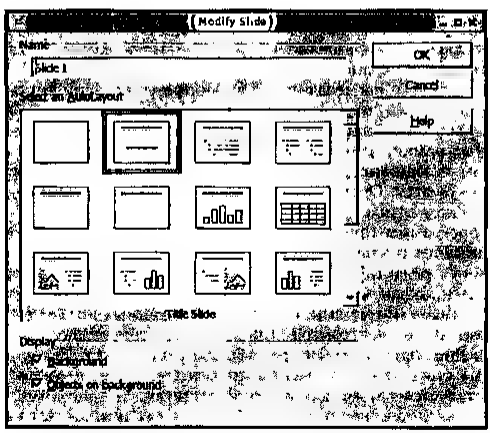


Рис 25 8 Программа Impress содержит ряд шаблонов слайдов

Закончив работу над первым слайдом, выберите команду Insert Slide в перемещаемом меню Presentation (расположено в левом нижнем углу на рис. 25.7). В результате снова появится диалоговое окно Modify Slide, с помощью которого выбирается структура очередного слайда. Процесс повторяется до тех пор, пока все слайды не будут созданы.

Для переключения между слайдами предусмотрены вкладки, расположенные в нижней части окна (Slide 1, Slide 2 и т.д. на рис. 25.7). Если нужно изменить порядок слайдов, выберите команду View⇨Master View⇨Slides View. Появится схема, описывающая порядок вызова слайдов, в которую разрешается вносить изменения.

Презентацию можно распечатывать точно так же, как и любой другой документ OpenOffice.org. Если компьютер оснащен очень большим экраном или адаптером диапроектора, запускайте презентацию непосредственно под управлением OpenOffice.org, так как это позволит использовать различные спецэффекты при смене слайдов. Для запуска презентации выберите команду Slide Show⇨Slide Show. После небольшой задержки экран очистится и появится первый слайд. Чтобы перейти к следующему слайду, выполните щелчок мышью. Если необходимо, чтобы слайды сменялись автоматически через определенные промежутки времени, вызовите окно настроек презентации с помощью команды Slide Show⇨Slide Show Settings и сделайте соответствующую установку.

Резюме

FreeBSD часто считают серверной операционной системой, и она действительно хорошо справляется с этой ролью. Однако во FreeBSD поддерживается также большое число офисных приложений, характерных для настольных компьютеров. Основными офисными пакетами являются GNOME Office, KOffice и OpenOffice.org. Существует и множество независимых программ, не относящихся к этим пакетам. В целом офисные пакеты FreeBSD не должны вызывать затруднений у пользователей других платформ, хотя некоторые операции, например обработка шрифтов, имеют свои особенности и заслуживают особого внимания.

Полный справочник по

Глава 26

Графические средства

Для кого-то графические редакторы — не более чем игрушка, а для кого-то — инструмент повседневной работы. Средства обработки графических изображений играют важную роль во многих областях. Вообще, такие средства довольно разнообразны. Существует два класса графических редакторов: одни работают с *растрами* (изображения, представляемые в виде совокупности точек определенного цвета), другие — с *векторной графикой* (изображения, создаваемые на основе линий, кривых и других графических примитивов).

В этой главе описывается несколько графических редакторов, начиная с наиболее мощного растрового редактора во FreeBSD: GIMP. Далее рассматриваются популярные программы векторной графики. В конце главы мы познакомимся с утилитами для обработки файлов формата PostScript и PDF. Такие файлы широко применяются для распространения электронных копий печатных страниц.

Программа GIMP

GIMP (GNU Image Manipulation Program — GNU-программа обработки изображений) является основным редактором растровой графики во FreeBSD. Эта программа считается официальной частью пакета GNOME Office, но она намного превосходит растровые редакторы других офисных пакетов. По своим возможностям она приближается к Adobe Photoshop. В Photoshop реализованы гораздо более удачные средства печати в формате CMYK (Cyan-Magenta-Yellow-Black — голубой-сиреневый-желтый-черный) и многие задачи программы выполняют по-разному, но в целом они сопоставимы.

Запуск программы

Для вызова GIMP введите `gimp` в терминальном окне. При первом запуске программа отображает серию диалоговых окон, позволяющих изменить настройки.

- **Персональный каталог GIMP**. Конфигурационные файлы GIMP хранятся в подкаталоге начального каталога пользователя. В GIMP 1.2 он называется `.gimp-1.2`. В нем содержится много других подкаталогов. Его имя поменять нельзя, и при запуске программа лишь сообщает о его содержимом.
- **Инсталляционный журнал**. Когда программа создает каталог `.gimp-1.2` и его подкаталоги, она отображает список выполняемых команд. Если в списке появятся сообщения об ошибках, щелкните на кнопке Cancel, чтобы прервать инсталляцию, найдите причину ошибки и устраните ее.
- **Параметры производительности**. Данные, с которыми работает программа, хранятся в кэш-буфере. По умолчанию его размер составляет 32 Мбайт, что вполне достаточно для однопользовательской рабочей станции с объемом оперативной памяти 128 Мбайт. Если это многопользовательская или интенсивно эксплуатируемая система, стоит уменьшить размер кэш-буфера. Можно также задать каталог, в который записываются данные, не помещаемые в кэш-буфере. Обычно это `.gimp-1.2`.
- **Разрешающая способность монитора**. Разрешающая способность важна, если требуется создавать изображения конкретного размера (в дюймах или сантиметрах). Она выражается в точках на дюйм (dpi — dots per inch). Замерьте горизонтальный и вертикальный размеры экрана и разделите каждое значение на горизонтальное и вертикальное разрешение в пикселях. Например, если на экране размером 11,5 на 8,7 дюймов выводится изображение размером 1024×768 пикселей, то горизонтальное разрешение составит 1024÷11,5=89 dpi, а вертикальное — 768÷8,7=88 dpi.

Когда программа запускается впервые, она заполняет каталог `.gimp-1.2` файлами, копируемыми из основной коллекции GIMP. Как долго продлится этот процесс, зависит от производительности компьютера. Далее появляется набор диалоговых окон (рис. 26.1).

Рис. 26.1. Программа GIMP использует целый ряд диалоговых окон, с помощью которых можно управлять работой программы

Совет В следующий раз программа запустится быстрее, поскольку ей не придется копировать стандартные файлы.

Обзор возможностей

В окне GIMP Tip of the Day отображается полезный совет, меняющийся при каждом следующем запуске программы. Если необходимости в этом окне нет, снимите флажок Show tip next time GIMP starts и щелкните на кнопке Close. Кнопки Previous Tip и Next Tip позволяют просмотреть советы.

Основное окно программы является самым маленьким из тех, что отображаются при запуске. На рис. 26.1 это верхнее окно справа. Остальные окна носят вспомогательный характер.

- Layers, Channels & Paths — позволяет манипулировать такими графическими элементами, как *слои*, *каналы* и *траектории*. В GIMP группы графических элементов можно объединять в слои, как если бы рисунки, выполненные на прозрачных листах, накладывались друг на друга. Такая методика дает возможность рисовать объекты независимо, а затем прятать их друг за другом, перемещать с переднего плана на задний и т.д. Каналы обозначают красные, зеленые и синие цветовые составляющие рисунков. С помощью траекторий можно создавать элементы векторной графики.
- Brush Selection — позволяет выбрать *кисть*, т.е. специальный узор, используемый при рисовании вручную с помощью мыши. Большинство кистей представляют собой круги различного радиуса, они предназначены для рисования

линий различной толщины Есть и более сложные кисти, применяемые для создания спецэффектов и сложных примитивов (добавляемых одним щелчком мыши)

- Tool Options. Содержимое этого окна меняется в зависимости от выбранного инструмента рисования. На рис 26.1 активизирован инструмент прямоугольного выделения, поэтому в окне представлены параметры касающиеся выделения областей изображений

Основное окно программы заполнено кнопками, каждая из которых соответствует определенному инструменту Инструменты предназначены для выполнения операций над изображениями При выборе инструмента содержимое окна Tool Options меняется При перемещении указателя мыши над изображением его вид тоже может меняться Если затем выполнить щелчок мышью будет выполнена операция характерная для выбранного инструмента.

Помимо набора кнопок в основном окне есть также строка меню, большинство элементов которой достаточно сложны и не рассматриваются здесь. Отдельного упоминания заслуживает команда Script-Fu, вызывающая одноименный инструмент. Его назначение — выполнять повторяющиеся действия над графическими файлами Это одно из достоинств GIMP

Загрузка и сохранение графических изображений

При запуске программа GIMP не загружает никаких файлов, если только имя файла не задано в командной строке Таким образом в первую очередь нужно создать новое изображение или загрузить существующее

Чтобы создать новый файл, выберите в меню File команду New. В результате появится диалоговое окно, в котором необходимо задать размер изображения в пикселях или дюймах. Можно также изменить разрешение экрана и указать, каким должно быть изображение: цветным или полутоновым. После щелчка на кнопке OK появится пустое окно рисунка

Если нужно загрузить существующее изображение, выберите в меню File команду Open и в появившемся диалоговом окне задайте нужный файл. Поддерживаются форматы TIFF, JPEG, PNG GIF, PSD и, разумеется собственный формат программы GIMP XCF Как правило, программа автоматически распознает тип файла, но можно принудительно задать тип в поле списка Determine File Type

Существуют различные подключаемые модули, позволяющие загружать изображения из нетрадиционных источников. Один из таких модулей является стандартным и доступен в меню File⇨Acquire⇨Screen Shot. Если выбрать этот модуль, программа отобразит диалоговое окно, содержащее команды захвата образа произвольного окна или всего экрана Можно также задать время задержки после которого будет активизирована команда захвата Если выбрана команда захвата окна, программа прочитает из экранного буфера образ первого окна, на котором пользователь выполнит щелчок иначе будет считан образ экрана Другим популярным модулем является утилита сканирования, позволяющая перехватывать сканируемые изображения

По завершении работы с изображением его нужно сохранить. Для этого выберите в меню File команду Save или Save As В первом случае файл сохраняется под первоначальным именем и в исходном формате Во втором случае можно задать другое имя файла или поменять его формат Важным элементом окна Save Image является поле списка Determine File Type По умолчанию в нем выбран элемент By Extension, означающий, что формат файла определяется по вводимому расширению. Например, файл сохраняется в формате XCF, если указано расширение `.xcf`, и в формате TIFF если указано расширение `.tif` или `tiff`

Современные форматы графических файлов предусматривают сжатие файла, что позволяет весьма существенно сократить его размер. Например, несжатое 24-разрядное изображение формата TIFF размером 500×500 пикселей занимает примерно 750 Кбайт, а сжатое — около 20 Кбайт. Если графический файл является частью Web-страницы, то уменьшение его размера позволяет ускорить загрузку страницы.

В некоторых графических форматах применяется сжатие без потери качества. Это означает, что в процессе загрузки файла можно восстановить исходное значение каждого бита. Такой режим является нормой в большинстве систем сжатия. В конце концов, вы же не хотите, чтобы при помещении файла в архив он повреждался? Но в случае графических изображений небольшая потеря информации вполне допустима. Это делается ради уменьшения размера файла. В идеале получается файл, неотличимый от оригинала при нормальном увеличении. В такой схеме, как правило, можно задавать коэффициент сжатия, и чем он выше, тем заметнее потеря качества.

В форматах TIFF, GIF, PNG и XCF сжатие либо не предусмотрено, либо выполняется без потери качества. JPEG — наиболее популярный формат, допускающий потерю качества.

На заметку

Не все форматы, распознаваемые программой при чтении файла, доступны для сохранения. Чаще всего проблема возникает с GIF-файлами, поскольку алгоритмы, используемые для создания таких файлов, защищены патентами. Программа GIMP не имеет права создавать файлы этого формата.

Если требуется создать сложное многослойное изображение, сохраните файл в формате XCF, чтобы записать в него информацию о слоях. По окончании редактирования такого файла его можно преобразовать в другой формат, например TIFF или JPEG. Это позволит просматривать файл в других программах. Но сначала нужно уплотнить изображение, т.е. свести все объекты в одном слое. Соответствующее преобразование выполняется с помощью команды Layers⇨Flatten Image. Если этого не сделать, в экспортированном файле будет лишь один исходный слой.

При сохранении файлов некоторых форматов необходимо задавать дополнительную информацию. Например, в формате JPEG поддерживаются различные уровни сжатия, устанавливаемые с помощью регулятора Quality в окне Save as JPEG. Поэкспериментируйте с различными установками и оцените результат. Аналогичным образом меняется степень сглаживания. Она влияет на обработку резких границ изображения в процессе сжатия.

Создание текстовых и графических элементов

Программа GIMP позволяет добавлять к изображению различные графические элементы, например текстовые строки, заполнять пустые области с использованием шаблонов заливки и т.д. В этом разделе мы поговорим о всевозможных способах модификации изображений с помощью инструментов главного окна.

Выбор цветов и шаблонов заливки

Когда в программе GIMP рисуется объект, ему ставится в соответствие два цвета: цвет переднего плана и цвет фона. Основным, "рабочим" цветом является первый. Именно он используется для рисования линий и заливки областей. Цвет фона учитывается лишь в некоторых операциях, например при градиентной заливке.

Для задания цвета переднего плана и цвета фона предназначен инструмент, расположенный в левом нижнем углу основного окна. На рис. 26.1 это область с изображением двух прямоугольников — черного и белого. Тот из них, который виден полностью, представляет цвет переднего плана, а тот, что позади — цвет фона. Чтобы изменить цвет, выполните двойной щелчок на нужном прямоугольнике. В результате появится окно Color Selection (рис. 26.2), в котором предусмотрены способы задания цвета.

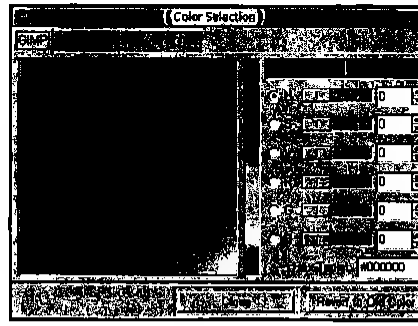


Рис 26.2 В программе GIMP предусмотрено несколько способов задания цвета

Инструмент, расположенный в правом нижнем углу окна, служит трем целям

- **Выбор кисти** В левом верхнем углу области представлен вид кисти, используемой в различных операциях рисования. На рис 26.1 это большой черный кружок. Если щелкнуть на этом фрагменте области, появится или переместится на передний план диалоговое окно Brush Selection (его можно увидеть на рис 26 1), в котором выбирается тип кисти
- **Выбор шаблона заливки** Справа от изображения кисти показан шаблон заливки В GIMP поддерживается множество стандартных шаблонов. Если щелкнуть на этом фрагменте области, появится или переместится на передний план диалоговое окно Pattern Selection, в котором можно задать тип шаблона Чтобы активизировать шаблон, необходимо выбрать параметр Pattern Fill в окне Tool Options
- **Выбор градиента** Некоторые инструменты рисования поддерживают использование градиентов, т.е. переходов заливки от фонового цвета к цвету переднего плана. Чтобы выбрать градиентный шаблон, щелкните на градиентной полоске в нижней части области Поддерживаются как плавные переходы, так и более экзотические шаблоны

Инструменты рисования

В программе GIMP есть несколько инструментов рисования. Большинство из них имитирует действие реальных инструментов художника, таких как карандаши и кисти Переместите указатель мыши на нужную кнопку основного окна программы, чтобы получить ее описание При щелчке на кнопке содержимое окна Tool Options меняется в соответствии с выбранным инструментом Многие инструменты должны быть знакомы пользователям других графических редакторов Назначение незнакомых инструментов несложно определить путем эксперимента

Заливка

Основной инструмент заливки представлен кнопкой, на которой изображено ведро с выливаемой краской. Можно осуществлять заливку цветом переднего плана, цветом фона или с использованием текущего шаблона. В процессе заливки фоновый цвет области, в которой выполняется щелчок мыши заменяется выбранным цветом или

шаблоном Если к примеру пользователь щелкает мышью внутри окружности полу чится круг, а область вне окружности останется без изменения

В программе есть также инструмент цветового перехода (Blend) позволяющий на кладывать градиент на изображение. Действие инструмента не ограничивается опре деленной областью, если только она не была предварительно выбрана Чтобы создать цветовой переход, выберите соответствующий инструмент и укажите две точки изо бражения. Они определят два крайних значения цветового диапазона. Обычно гради ент полностью замещает изображение но с помощью регулятора Opacity в окне Tool Options можно уменьшать непрозрачность градиента позволяя существующему изо бражению "просвечивать" сквозь него.

Добавление текстовых надписей

В программе имеется текстовый инструмент позволяющий добавлять к изображе нию текстовые надписи Этот инструмент обозначен буквой 'T'. Если, выбрав его щелкнуть в окне изображения появится диалоговое окно Text Tool (рис. 26 3). Введите текст в поле Preview и задайте шрифт, начертание и размер текста После щелчка на кнопке OK надпись будет добавлена к изображению.

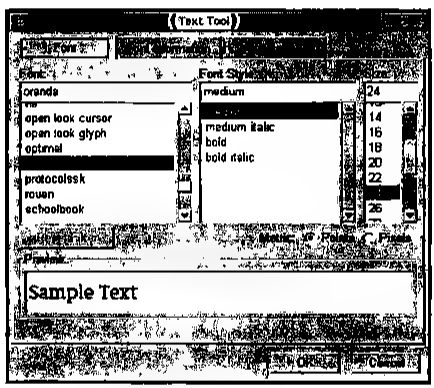


Рис 26 3 Для создания надписей предназначено отдель ное диалоговое окно с возможностью предварительного просмотра текста

Первоначально текст добавляется в отдельный слой, что упрощает его позициони рование. Для перемещения текста предназначен инструмент Move Layers and Selection (кнопка с изображением четырех стрелок). Щелкните на текстовом объекте и, удер живая кнопку мыши, измените его положение

Преобразования

Добавление элементов к изображению — важная операция, но часто требуется преобразовать существующее изображение а не дополнять его. В программе GIMP есть много средств преобразования, позволяющих усекать изображения, менять их цветовую палитру и применять фильтры

Выборка и усечение изображений

Первый ряд инструментов в основном окне программы — это все инструменты выборки областей. прямоугольных, округлых, произвольной формы и др Над вы бранной областью можно выполнять различные операции, а именно

- перемещение (щелкните мышью в пределах области и не отпуская кнопку перетащите область в нужное место),
- копирование (нажмите <Ctrl+C>),
- вырезание (нажмите <Ctrl+X>)
- применение стандартных инструментов например градиента только к выде ленному участку;
- применение фильтра только к выделенному участку

Похожей операцией является *усечение* изображения, т е удаление всего кроме вы бранной области Чтобы выполнить усечение, щелкните на изображении правой кнопкой мыши и выберите команду Tools⇨Transform Tools⇨Crop & Resize или же вос пользуйтесь соответствующим инструментом (кнопка с изображением скальпеля) За тем щелкните левой кнопкой мыши в одном из углов выделяемой области и, не от пуская кнопку, перетащите указатель в противоположный угол, после чего отпустите кнопку. Программа отобразит диалоговое окно Crop & Resize, в котором можно задать параметры области. Кнопка Crop позволяет удалить все, что расположено за предела ми области Кнопка Resize предназначена для сжатия всего изображения до размеров выделенной области

Изменение цветов

В программе GIMP есть несколько инструментов, предназначенных для измене ния цветов изображения Они наиболее полезны при обработке оцифрованных фо тографий Можно менять яркость и контрастность изображения уровни насыщенно сти цветов, выполнять более специфические трансформации Большинство инстру ментов доступно в меню Image⇨Colors, вызываемом из контекстного меню

Стоит сказать несколько слов о такой важной операции, как *индексирование* цве тов В некоторых графических форматах поддерживается очень ограниченное количе ство цветов, и даже в более универсальных форматах полезно сокращать число ис пользуемых цветов, так как это позволяет уменьшить размер файла Чтобы активизи ровать инструмент индексации, выберите в контекстном меню команду Image⇨ Mode⇨Indexed Количество цветов, которое должно остаться после индексации зада ется пользователем (по умолчанию — 256, т.е. 8-разрядная трансформация)

Применение фильтров

В GIMP содержится множество *фильтров*, которые манипулируют изображением определенным образом. Фильтры доступны из контекстного меню Filters и разделены на категории, например Blur, Noise, Glass Effects и Artistic. Одни фильтры добавляют к изображению новый элемент, такой как фон, другие выполняют графическое преоб разование, моделируя заданный спецэффект.

Программы векторной графики

Редакторы векторной графики работают с математически определенными фигура ми а не с растрами. Это позволяет создавать изображения, состоящие не из отдель ных пикселей а из виртуальных объектов, к тому же хорошо масштабирующиеся на

любом устройстве вывода. В программе GIMP есть слои, на которых можно создавать независимо перемещаемые объекты, но это не решает проблему масштабирования. Векторные редакторы определяют изображения в терминах конкретных геометрических объектов, таких как окружности и линии, перемещаемых независимо. Спектр таких программ достаточно широк. В этом разделе будут рассмотрены основные программы, доступные во FreeBSD.

На заметку Векторный редактор Draw, являющийся частью пакета OpenOffice.org, рассматривался в главе 25.

Xfig

Xfig (`www.xfig.org`) — простой векторный редактор общего назначения. В нем можно создавать множество простейших диаграмм и рисунков. Для запуска программы введите **`xfig`** в терминальном окне. На рис. 26.4 представлено окно программы. Как нетрудно заметить, программа позволяет создавать линии различной длины, толщины и формы, а также целый ряд других геометрических фигур (прямоугольники, окружности, многоугольники, кривые и т.д.). Поддерживается создание текстовых надписей. Несколько объектов можно сгруппировать, образовав более сложную фигуру. В программе есть целая библиотека графических заготовок. Например, изображения компьютеров на рис. 26.4 взяты из библиотеки.

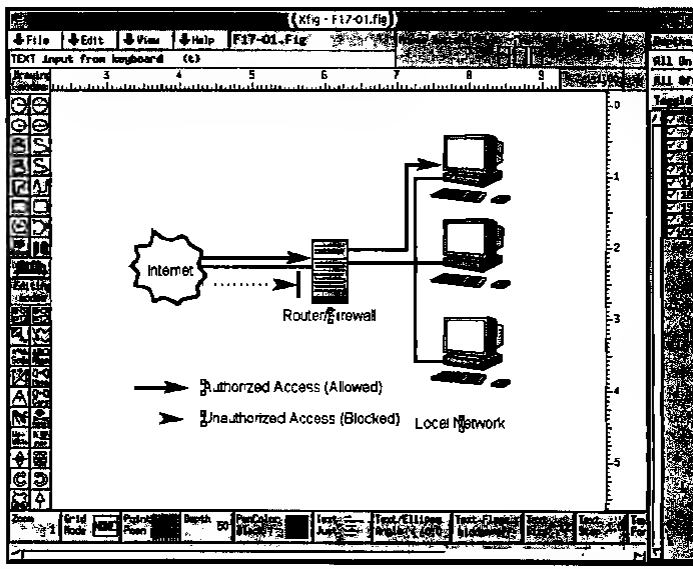


Рис. 26.4. Xfig — типичный векторный редактор общего назначения

Сразу после запуска программы область рисования пуста. Чтобы начать рисовать щелкните на одной из кнопок панели инструментов в левой части окна. Верхняя половина панели — это всевозможные фигуры: окружности, эллипсы, кривые, многоугольники и т.д. Здесь же находятся кнопки для вставки текста, растровых изображений и библиотечных заготовок. Как правило, при щелчке мышью в области рисования устанавливается угловая, начальная или центральная точка объекта. Нужно переместить указатель мыши и снова выполнить щелчок, чтобы определить противоположный угол, внешний радиус или конечную точку. Некоторые объекты, например многоугольники, могут содержать произвольное число управляющих точек. По окончании ввода точек щелкните средней кнопкой мыши (для некоторых типов объектов программа Xfig проводит линию от последней точки к первой, создавая замкнутую фигуру).

У большинства объектов есть несколько параметров, таких как толщина и форма линий (сплошные, пунктирные и т.д.), цвет линий, цвет и стиль заливки. Для текстовых объектов задается тип и размер шрифта. Все эти параметры доступны с помощью кнопок, расположенных в нижней части окна (на рис. 26.4 представлены параметры текстового объекта). Одним из параметров является *глубина* объекта. Эта характеристика определяет позицию объекта в "стеке" перекрывающихся элементов изображения. Объект с меньшей глубиной перекроет объект с большей глубиной, если их разместить друг на друге.

Для модификации созданных объектов предназначены кнопки в нижней половине панели инструментов. Можно сгруппировать несколько объектов в один метаобъект, разбить существующую группировку, изменить масштаб объекта, скопировать или удалить объект, изменить его характеристики, повернуть или зеркально отобразить объект и т.д. В большинстве случаев соответствующее действие выполняется немедленно, но иногда программа Xfig отображает диалоговое окно, в котором нужно ввести дополнительную информацию.

Диалоговые окна программы Xfig иногда содержат поля ввода текста, которые ведут себя немного странно: при вводе текста в такое поле указатель мыши должен быть наведен на него, иначе поле будет недоступно.

У программы Xfig есть небольшая строка меню, содержащая типичные файловые команды и команды редактирования. Рисунки можно сохранять в собственном формате программы или экспортировать в формате PostScript, EPS, TIFF и др. В меню Edit доступны команды отмены, вырезания/вставки и т.д. Команды меню Zoom позволяют масштабировать изображение, отображать число вершин и т.д. Из меню Help вызываются файлы справки. Некоторые компоненты редактора зависят от внешних программ, таких как Web-броузер или утилита проверки орфографии. Соответствующие параметры задаются в меню Edit⇨Global Settings.

Dia

Dia (`http://www.gnome.org/gnome-office/dia.shtml`) — это более специализированный векторный редактор, являющийся частью пакета GNOME Office и предназначенный для создания диаграмм. Для запуска программы введите **dia** в терминальном окне. Появится небольшое управляющее окно. Выберите в нем команду File⇨New Diagram, чтобы отобразить окно рисования. Оба окна изображены на рис. 26.5.

В программе Dia предусмотрено меньше типов объектов, чем в Xfig, хотя у некоторых объектов есть параметры, позволяющие им выполнять те же функции, что и объекты Xfig. Например, в программе Xfig существуют отдельные объекты для прямо

угольников с прямыми и скругленными углами тогда как в Dia это один объект с параметром, определяющим радиус скругления В нижней части управляющего окна расположены инструменты, позволяющие задавать цвет объекта толщину и стиль линий (сплошные, пунктирные и т.д), а также их тип (обычные, со стрелкой на конце и др.) Инструмент выбора цвета работает так же, как и в программе GIMP: щелкните на изображении соответствующего прямоугольника чтобы задать цвет фона или переднего плана

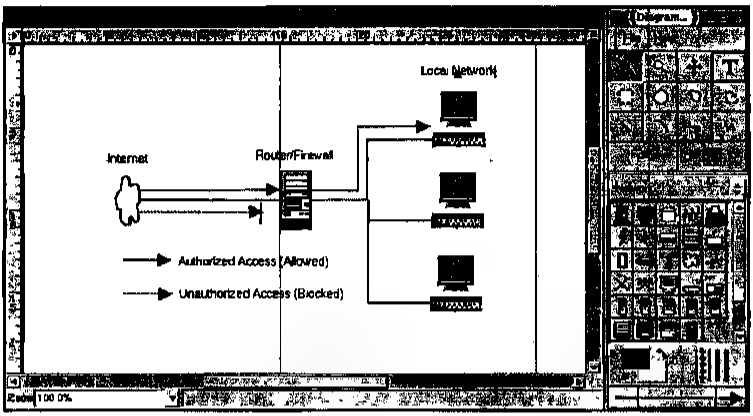


Рис 26 5 Назначение элементов управления программы Dia — размещение стандартных объектов

Главным достоинством Dia является набор готовых объектов Они объединены в группы, например Circuit, Flowchart, Logic и Network Меню выбора группы находится в средней части управляющего окна (на рис. 26 5 выбрана группа Network). При выборе группы под ней отображаются кнопки доступных объектов. Чтобы добавить объект к рисунку выберите его и щелкните в окне документа. В этом месте программа и вставит объект. Если щелкнуть на объекте правой кнопкой мыши, появится меню со списком допустимых операций над объектом. Некоторые действия например выравнивание, возможны, только когда выбрано несколько объектов.

Чтобы сохранить файл Dia, щелкните правой кнопкой мыши в окне документа и выберите команду File⇨Save или File⇨Save As. Команда File⇨Export предназначена для экспорта фойла в другом формате, например EPS или PNG.

Выбор некоторых форматов экспорта приводит к потере информации Например, если выбрать формат Xfig, то не исключено что в программе Xfig объекты диаграммы будут искажены

Gnuplot

Gnuplot (`www gnuplot info`) — это текстовая утилита построения графиков. Такое определение кажется парадоксальным, но лишь на первый взгляд программа чи тает файл, содержащий числовые данные, и создает графический файл предназначенный для вывода данных в виде графика Можно непосредственно отобразить этот

файл в X-окне или создать традиционный файл для печати или редактирования в текстовом процессоре

На заметку Несмотря на название, программа Gnuplot не связана с проектом GNU. Она даже не сопровождается лицензией GPL, хотя и является бесплатной

Описание программы Gnuplot лучше начать с примера. В листинге 26.1 показан файл данных, для которых строится график. Данные представлены в текстовом виде (такие файлы импортируются большинством редакторов электронных таблиц и другими утилитами анализа данных). В первой колонке содержатся значения по оси X, во второй — по оси Y. Здесь представлена средняя стоимость изделий некой компании Widget Company, изменяющаяся с каждым годом.

Листинг 26.1. Образец входного файла программы Gnuplot (`data.txt`)

```
1995 26.34
1996 24.97
1997 25.02
1998 25.67
1999 24.78
2000 23.02
2001 23.69
2002 23.32
```

Чтобы по этим данным построить график, выполните следующие действия.

1. Введите данные, показанные в листинге 26.1, в текстовый файл `data.txt`.
2. Создайте управляющий файл программы. В нем должны содержаться все параметры, определяющие вид графика. Образец такого файла, названного `plot.ctl`, показан в листинге 26.2.
3. Введите `gnuplot -persist plot.ctl`. Эта команда создает X-окно, в котором отображается график (рис. 26.6).

Листинг 26.2. Образец управляющего файла программы Gnuplot (`plot.ctl`)

```
set title 'Widget Cost Over Time'
set xlabel "Year"
set ylabel "Cost"
set output
set terminal x11
plot [1995:2002] [22:28]  data.txt' with lines
```

Команды программы, показанные в листинге 26.2, можно задавать и в интерактивном режиме. Для перехода в интерактивный режим (ему соответствует строка приглашения `gnuplot>`) введите просто `gnuplot` вместо `gnuplot -persist plot.ctl`.

Большинство команд управляющего файла задает различные параметры программы. В первых трех строках листинга 26.2 определяются надписи, такие как заголовок графика и подписи осей X—Y. Далее устанавливаются параметры вывода. Строка `set output` — это "заглушка", она не нужна, если данные выводятся в X-окне. Команда `set terminal` задает тип выходного файла. В листинге 26.2 это X-окно.

Если необходимо создать графический файл, предназначенный для печати или импортирования в другое приложение, измените параметры `set output` и `set terminal`. В первом случае требуется указать имя файла в кавычках, например `set output "plot.fig"`. Во втором случае вместо `x11` может быть указано несколько типов файлов.

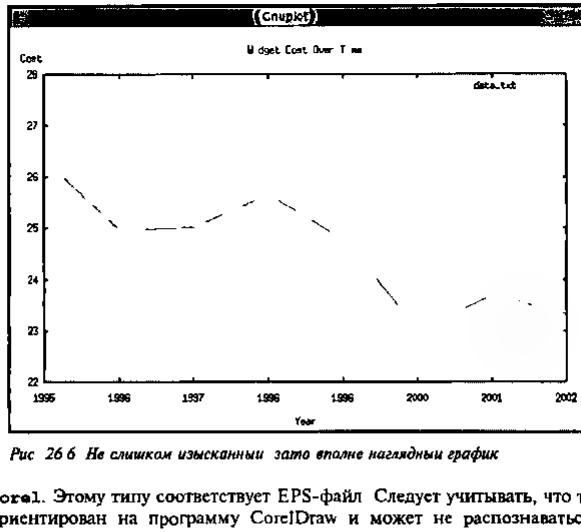


Рис 26 6 Не слишком изысканный зато вполне наглядный график

- **corel**. Этому типу соответствует EPS-файл Следует учитывать, что такой файл ориентирован на программу CorelDraw и может не распознаваться другими программами.
- **fig** Этому типу соответствует файл, импортируемый программой Xfig Программа сообщит о том, что файл представлен в ctrom формате, но тем не менее корректно его загрузит.
- **hpgl** HPGL (Hewlett-Packard Graphics Language) — формат векторной графики непосредственно распознаваемый многими лазерными принтерами семейства Hewlett-Packard
- **pcl5** PCL (Printer Control Language) — язык управления принтером, используемый лазерными принтерами семейства Hewlett-Packard. В данном случае формат файла является растровым
- **postscript**. Этот тип позволяет непосредственно распечатать файл поскольку FreeBSD считает все принтеры PostScript совместимыми
- **latex** В этом случае будет создан файл который можно включить в документ LaTeX
- **png** Этому типу соответствует растровый PNG-файл Такой файл можно редактировать в программе GIMP или включить в документ текстового процессора

Существует много других типов, соответствующих различным моделям принтеров Чтобы получить список доступных опций вывода введите **set terminal** в командной строке программы Gnuplot

Наиболее важной командой листинге 26 2 является следующая

```
plot [1995:2002] [22.28] 'data.txt' with lines
```

Она заставляет программу Gnuplot построить график данных, содержащихся в файле `data.txt`. Две группы чисел в квадратных скобках представляют шкалы значений для осей X и Y Если их не указать программа автоматически выберет масштаб

осей, построив график охватывающий все исходные значения Аргумент `with lines` говорит о том, что точки графика должны быть соединены линиями. Вместо `lines` могут стоять другие параметры

- **`points`** каждая точка графика представляется отдельным ромбиком;
- **`linespoints`** — точки графика соединяются линиями, но при этом дополни тельно выделяются·
- **`dots`** — аналог опции `points`, но точки графика имеют гораздо меньшие раз меры, что удобно, если точек очень много
- **`impulses`** — точки графика соединяются линиями с осью X
- **`boxes`** — создается гистограмма, между столбиками которой нет промежутков;
- **`steps`** — в определенном смысле это аналог опции `linespoints` но соедини тельные линии могут быть либо горизонтальными, либо вертикальными
- **`xerrorbars`** и **`yerrorbars`** — эти опции задают вывод, соответственно гори зонтальных и вертикальных планок погрешностей Обе они требуют наличия третьей колонки данных во входном файле в которой указывается величина погрешности

Чтобы получить полный список типов графиков, введите **`plot`** в командной строке программы, указав фиктивное значение параметра `with` В ответ программа отобразит список поддерживаемых вариантов.

Если необходимо построить сразу несколько графиков, укажите имена соответет вующих входных файлов в одной строке `plot`, например

```
plot [1995.2002] [22:28] 'data.txt' with lines, 'data2.txt  with lines
```

Программа Gnuplot является чрезвычайно эффективным средством визуализации данных Она способна создавать даже трехмерные графики. Дополнительную инфор мацию можно найти во внутренней справочной системе (чтобы вызвать ее, введите **`help`** в командной строке) а также на Web-узле программы.

Обработка файлов PostScript, EPS и PDF

В главе 9, где описывалась подсистема печати во FreeBSD, говорилось о том, что FreeBSD-программы предполагают совместимость принтеров со стандартом PostScript. Однако PostScript — это не только язык управления печатью, но и полноценный язык программирования, а также формат графических файлов. На последнем утверждении следует задержаться, так как этот формат очень важен Многие FreeBSD-программы выводящие данные на печать, способны создавать выводные файлы в формате PostScript Такие файлы можно печатать вручную просматривать с помощью специ альных программ, преобразовывать в другой формат с помощью программы Ghostscript и перемещать на другой компьютер, где с ними можно делать то же самое

С форматом PostScript связаны два других распространенных файловых формата Первый из них, EPS (Encapsulated PostScript), напоминает PostScript, но предназначен для описания небольших графических элементов (размером меньше страницы), кото рые можно встраивать в PostScript-документы. EPS-файлы содержат как графику, так и текст. Многие программы умеют создавать EPS-данные или встраивать EPS-файлы в собственные документы. Программа Ghostscript обрабатывает EPS-файлы так же как и файлы PostScript. Второй родственный формат файлов разработан компанией Adobe и называется PDF (Portable Document Format — формат переносимых докумен тов) PDF-файлы еще называют файлами Acrobat, поскольку они создаются и считы ваются одноименной программой С PDF файлами чаще работают на других плат

формах, например в Windows, где пакет Ghostscript и другие средства обработки PostScript-файлов, как правило, не инсталлированы. PDF-файлы обычно имеют меньший размер чем их PostScript-аналоги

Программа Ghostscript

Ghostscript — это наиболее удобное средство манипулирования PostScript-файлами Чаще всего программа используется в качестве обработчика очереди на печать, как описывалось в главе 9, однако ее можно применять и для преобразования файлов формата PostScript, EPS иди PDF в другие форматы. Именно так собственно говоря, и осуществляется обработка очереди на печать, просто в этом случае формат выход ного файла соответствует формату входных данных принтера Например, для многих лазерных принтеров это PCL.

Предположим, имеется PostScript-файл Возможно, он был сгенерирован локаль ной программой или загружен из Internet. Вот как преобразовать его в другой формат с помощью программы Ghostscript

```
$ gs -dNOPAUSE -dBATCH -dSAFER  r72x72  sDEVICE=jpeg ↳
sOUTPUTFILE=file.jpg file.ps
```

Назначение аргументов команды таково

- **-dNOPAUSE** — запрещает программе делать паузу между страницами. По умол чанию программа запрашивает у пользователя разрешение на создание каждой следующей страницы выходных данных, что весьма неудобно в случае боль шого документа
- **dBATCH** — заставляет программу завершиться сразу по окончании обработки задания В противном случае программа отобразит строку приглашения GS> и будет ожидать ввода команд.
- **-dSAFER**. Поскольку PostScript язык программирования, в виде PostScript файла вполне может распространяться злонамеренный программный код Опция -dSAFER запрещает программе удалять и переименовывать файлы, а также открывать их для записи, что сводит риск к минимуму, существенно сокращая возможности троянских коней, маскирующихся под PostScript-файлы
- **-r72x72**. Многие действия программы связаны с преобразованием изображе ния из растрового формата (PostScript) в пиксельный. В традиционном PostScript-файле расстояния определены в физических единицах, тогда как фи зические размеры пиксельного изображения зависят от разрешающей способ ности выходного устройства Опция -r позволяет задать разрешающую спо собность в точках на дюйм. Последующие значения разделенные символом x интерпретируются как горизонтальное и вертикальное разрешение. Одиночное значение говорит о том, что оба разрешения одинаковы Например, выражение r72 эквивалентно записи -r72x72
- **sDEVICE=jpeg** — задает выходной формат в данном случае JPEG Поддержи ваемые форматы будут описаны ниже
- **-sOUTPUTFILE=file.jpg** задает имя выходного файла. Если этот аргумент опустить, программа будет подразумевать стандартное выходное устройство что приведет к "засорению" экрана консоли или окна
- **file.ps**. В конце задается имя входного файла. Если программа Ghostscript вызывается из сценария, вместо входного файла часто указывается дефис обо значающий считывание данных из стандартного входного потока

Входные файлы могут быть представлены в формате PostScript EPS или PDF. Что бы получить список выходных форматов, введите `gs --help`. Рассмотрим наиболее [illegible]

- **BMP**. Файлы формата BMP имеют расширение `bmp`. Программа Ghostscript поддерживает несколько вариантов попиксельного вывода, включая `bmpmono`, `bmpgray`, `bmp16`, `bmp256` и `bmp16m`. Они различаются типом и размером цветовой палитры.
- **PCX**. Файлы формата PCX изначально использовались программой Paintbrush. Они имеют расширение `.pcx`. Программа Ghostscript поддерживает следующие разновидности формата: `pcxmono`, `pcxgray`, `pcx16`, `pcx256`, `pcx24b` и `pcxcmyk`. Как и в случае формата BMP, здесь создаются файлы с различными вариантами цветовой палитры.
- **TIFF**. Для формата TIFF поддерживаются режимы `tiff12nc`, `tiff24nc`, `tifflzw` и `tiffpack`. Последние два разрешают сжатие данных.
- **PNG**. Для формата PNG поддерживаются режимы `pngmono`, `png16`, `png256` и `png16m`. Они различаются типом и размером цветовой палитры.
- **JPEG**. У формата JPEG есть лишь две разновидности: `jpeg` и `jpeggray`. В первом случае создаются цветные изображения, во втором — полутоновые.
- **PostScript**. Формату PostScript соответствуют режимы `psmono`, `psgray`, `psrgb` и `pswrite`.
- **EPS**. В случае опции `epswrite` создается EPS-файл.
- **PDF**. В случае опции `pdfwrite` создается PDF-файл.

Это выглядит странно — создавать PostScript-файлы из файлов аналогичного типа, но такая возможность весьма полезна. Выходной PostScript-файл является растровым и будет обработан принтером иначе, чем исходный файл. Таким образом, если исходный файл слишком сложен или содержит трудно обнаруживаемые ошибки, вызывающие проблемы при печати с помощью программы Ghostscript можно выполнить предварительную обработку файла, упростив задачу принтеру с ограниченным объемом памяти или некорректно работающим интерпретатором PostScript.

В большинстве случаев выходные файлы программы Ghostscript представляются в пиксельном виде. Исключение составляют PDF файлы, которые содержат шрифты и относятся к векторному типу. PDF — это популярный формат распространения текстовых файлов в электронном виде, и программа Ghostscript является удобным средством создания таких файлов. "Распечатайте" текстовый документ в PostScript-файл, передайте полученный результат программе Ghostscript — и вы получите PDF-файл, который может быть прочитан программами просмотра на любой платформе.

Графические утилиты просмотра PostScript-файлов

Иногда нужно просмотреть PostScript-файл прежде чем выводить его на печать. Кому хочется выбросить в мусорную корзину сотню испорченных страниц? Программа Ghostscript позволяет предварительно просмотреть документ непосредственно или с помощью внешней надстройки. В простейшем случае достаточно ввести команду `gs`:

```
$ gs -dBATCH -dSAFER -sDEVICE=x11 file.ps
```

В результате появится X-окно, в котором отобразится документ, а в терминальном окне программа Ghostscript выведет следующее приглашение:

```
>>showpage, press <return> to continue<<
```

При нажатии клавиши <Enter> будет выведена следующая страница документа. По достижении последней страницы программа завершит работу. Такой метод просмотра приемлем лишь для небольших документов. Существуют более удобные графические надстройки к программе Ghostscript. Рассмотрим основные из них:

- **Ghostview** — простейшая надстройка к Ghostscript. Для вызова программы введите `ghostview` с указанием имени нужного файла. Интерфейс программы довольно примитивен по современным стандартам.
- **gv** — программа, созданная на основе Ghostview. Она имеет большее число опций и более современный интерфейс.
- **GNOME Ghostview** — эта программа распространяется в составе оболочки GNOME и по функциональным возможностям напоминает `gv`. Исполняемый файл называется `ggv`.

Все эти программы доступны в разделе `printing` коллекции пакетов или коллекции переносимых приложений (каталоги `ghostview`, `gv` и `ggv`). Все программы вызываются одинаково: достаточно указать имя программы и загружаемый файл. Кроме того, все они отображают файлы с использованием программы Ghostscript, поэтому результаты очень похожи. Программы GNOME Ghostview и `gv` располагают рядом расширенных возможностей. Например, они поддерживают сглаживание шрифтов, что в некоторых случаях позволяет повысить разборчивость текста. Все программы могут отображать файлы форматов PostScript, EPS и PDF, хотя не все элементы PDF-файлов воспроизводятся корректно.

На рис. 26.7 приведено окно GNOME Ghostview, в котором отображается PostScript-файл. В левой части окна расположены элементы управления, в том числе список страниц. Для перемещения по документу достаточно щелкнуть на номере страницы. Эта возможность впрочем, доступна не для всех документов. Иногда допускается лишь постраничное перемещение вперед-назад. Можно также увеличивать или уменьшать изображение, используя кнопки с изображением лупы, менять ориентацию виртуальной страницы и включать/отключать режим сглаживания шрифтов. Программа разрешает напечатать весь документ или выбранную группу страниц.

Графические утилиты просмотра PDF-файлов

Как указывалось выше, распространенные утилиты просмотра PostScript-файлов распознают также файлы форматов EPS и PDF. Но есть ряд специализированных утилит, работающих только с PDF-файлами. Среди них выделим две:

- **Xpdf** — программа с открытым исходным кодом, работающая в X-среде. Ее функциональные возможности минимальны. Основную часть окна программы занимает область просмотра документа, строка меню отсутствует, есть лишь несколько кнопок в нижней части окна, предназначенных для навигации, поиска текста и печати файла.
- **Acrobat Reader** — программа компании Adobe, считающаяся эталоном, поскольку эта компания является разработчиком формата PDF (как и самого языка PostScript). Соответствующий пакет называется `acroread4`, но с технической точки зрения это не FreeBSD-программа, а Linux-программа, для работы которой необходимы средства бинарной совместимости с Linux, рассматривавшиеся в главе 4.

В целом указанные программы работают примерно так же, как и программы просмотра PostScript-файлов, хотя многие детали, конечно же, отличаются. Возможности Acrobat Reader шире в том плане, что эта программа способна отображать некоторые PDF-файлы, недоступные другим программам просмотра. Проблемные файлы созда-

ются с помощью последних версий программы Acrobat Distiller. PDF-файлы, созданные программой Ghostscript, редко вызывают проблемы. Ряд специфических элементов PDF, таких как таблицы содержимого и некоторые типы встроенных объектов, тоже распознаются в полном объеме только в Acrobat Reader. Если планируется регулярное чтение PDF-файлов, особенно если эти файлы поступают из разных источников, инсталлируйте пакет `acroread4` и библиотеки совместимости с Linux.

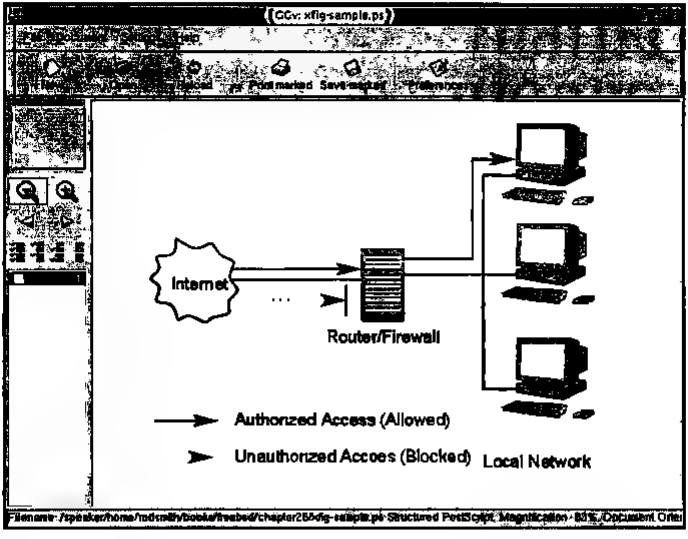
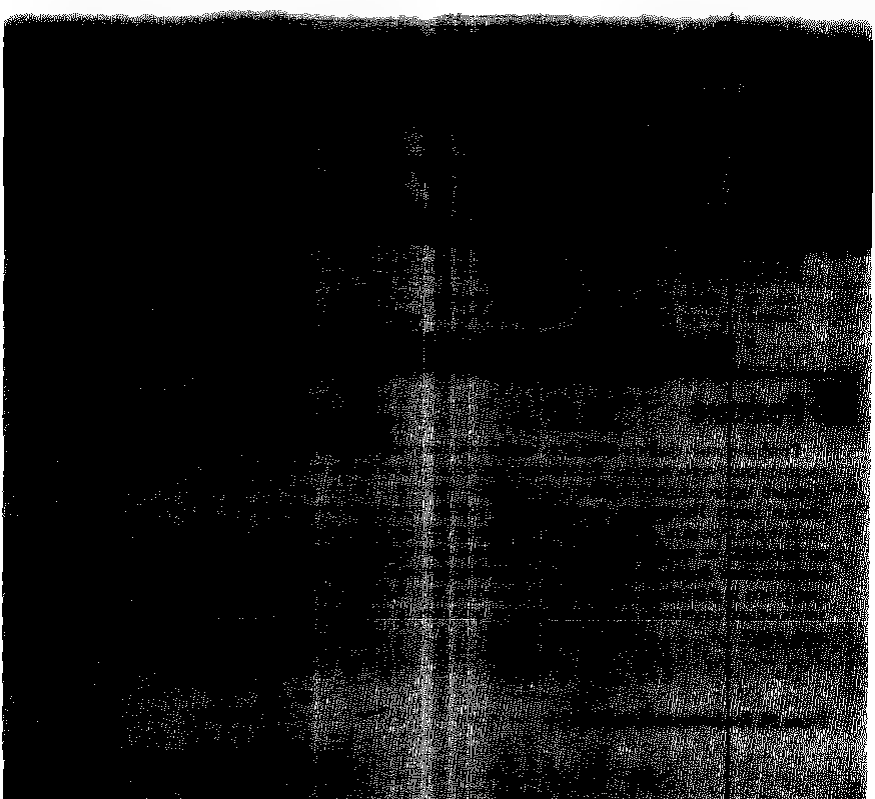


Рис. 26.7. Программа GNOME Ghostview отображает PostScript-документ, при этом пользователь имеет возможность перемещаться по документу и менять параметры просмотра

Резюме

Графические файлы бывают множества видов, и для работы с ними требуются специальные программы. Различают два класса изображений: растровые и векторные. Первые представляют собой двухмерные массивы пикселей, с каждым из которых связан цвет. Растровые изображения плохо масштабируются, зато с их помощью удобно кодировать фотографии. Основная программа обработки таких изображений во FreeBSD называется GIMP. Векторные изображения задаются в виде набора геометрических форм. Они хорошо масштабируются и удобны для создания разного рода диаграмм и рисунков. Существует большое число редакторов векторной графики, позволяющих создавать и просматривать векторные изображения, а также манипулировать ими.

Полный справочник по



Сегодня почти все компьютеры оснащены звуковой платой и высокопроизводительной видеоплатой. Оба этих аппаратных компонента широко используются в мультимедийных и игровых приложениях. Примерами мультимедийных приложений являются проигрыватели видеоклипов, системы видеоконференций и программы редактирования цифровых фильмов. FreeBSD нельзя назвать оптимальной мультимедийной или игровой платформой, но соответствующие программные средства в ней есть. Именно их мы и рассмотрим в настоящей главе.

В первую очередь читатели познакомятся с аппаратными и программными средствами обработки звука. Видеоподсистема (представленная средой X Window System) является частью стандартной мультимедийной конфигурации FreeBSD на рабочей станции. А вот аудиосредства гораздо реже бывают правильно сконфигурированы. Просто они не нужны для обработки текста, редактирования графики и выполнения множества других распространенных задач. Далее будут рассмотрены программы воспроизведения цифровых фильмов и анимации, а также некоторые из доступных игр.

На заметку Большинство из описываемых в этой главе программ может быть инсталлировано из раздела `audio` коллекции пакетов или коллекции переносимых приложений.

Обработка звука во FreeBSD

Современные мультимедийные приложения имеют дело со звуковыми и визуальными данными. Обработку последних берет на себя X Window System, хотя иногда это осуществляется с помощью библиотеки *SVGAlib*, которая позволяет программам получать доступ к различным видеорежимам SVGA в отсутствие X-среды. Поскольку X Window System уже рассматривалась в главе 13, в этом разделе мы поговорим об обработке звука — какие средства для этого требуются и как их конфигурировать. Поддержка звука реализуется драйверами ядра FreeBSD для конкретных режимов звуковой платы.

На заметку Компьютеры семейства x86 оснащены простейшим звуковым оборудованием, включающим громкоговоритель, установленный на внутренней стороне корпуса, и звуковой генератор на материнской плате. Этих устройств достаточно для воспроизведения звука, однако его качество будет минимальным. Для получения качественного звука необходимы звуковая плата и внешние колонки. Кроме того, у звуковой платы есть входные порты для записи звука из различных источников. При наличии одного лишь внутреннего динамика используется драйвер `snd_pca.ko`.

Поддерживаемое звуковое оборудование

Чтобы выяснить, поддерживается ли звуковая плата, обратитесь к файлу `HARDWARE.TXT`, поставляемому вместе с FreeBSD (он находится в корневом каталоге инсталляционного компакт-диска). В разделе "Audio Devices" перечислены поддерживаемые микропроцессорные наборы звуковых плат. Естественно, список выглядит неполным, поскольку один и тот же набор может использоваться в различных устройствах. Например, микропроцессорные наборы Crystal Semiconductor, ESS и OPTi встречаются в самых разных звуковых платах. Драйверы Intel и VIA распознают встроенные звуковые микросхемы материнских плат.

На заметку Если у материнской платы есть встроенные звуковые микросхемы, не распознаваемые во FreeBSD, можно просто установить обычную звуковую плату и в дальнейшем игнорировать присутствие встроенного устройства. Чтобы избежать конфликтов, отключите встроенную звуковую поддержку через BIOS

Скомпилированные звуковые модули ядра содержатся в каталоге */boot/kernel* (FreeBSD 5.0) или */modules* (FreeBSD 4.x). Имена файлов начинаются с `snd`, поэтому для получения списка модулей нужно ввести **`ls /boot/kernel/snd*`** (**`ls /modules/snd*`** во FreeBSD 4.x). Если же в процессе перекомпиляции ядра в него были интегрированы средства поддержки звука, то модулей в указанных каталогах не будет.

К сожалению, определить нужный драйвер не всегда легко. Необходимо решить две проблемы:

- **узнать имеющийся набор звуковых микросхем.** Если на компьютере инсталлирована Windows, загрузитесь в ней и просмотрите информацию о звуковом устройстве, в противном случае просмотрите надписи на самой плате. Определенные подсказки может дать команда `dmesg`, если FreeBSD автоматически обнаруживает плату.
- **выяснить, какой драйвер соответствует набору микросхем.** В списке микропроцессорных наборов, содержащемся в файле `HARDWARE.TXT`, указано имя драйвера для каждого из наборов, например `snd_ess.ko` для ESS или `snd_via82c686.ko` для VIA.

Если определить драйвер не удалось, протестируйте имеющиеся модули, чтобы выяснить, не подойдет ли один из них. Некорректный звуковой модуль можно загрузить, просто он не будет работать.

Совет В качестве последнего шанса, если драйвер для звуковой платы найти не удалось, обратитесь на Web-узел компании 4Front Technologies (`www.opensound.com`). Она выпускает коммерческие звуковые драйверы для FreeBSD и других операционных систем семейства UNIX. Загрузите демо-версию драйвера и, если она подойдет, заплатите регистрационный взнос: ведь он наверняка будет меньше, чем стоимость новой звуковой платы.

Запуск звуковой подсистемы

Загрузив систему, введите **`dmesg | less`**, чтобы просмотреть сообщения ядра. Если ядро скомпилировано с поддержкой звуковой платы, должно появиться одно или несколько сообщений, идентифицирующих плату. К примеру, следующие сообщения соответствуют плате Gravis UltraSound (GUS) PnP:

```
gusc0: <Gravis UltraSound Plug & Play PCM> at port 0x32c 0x32f ↵
0x320-0x327, 0x220-0x22f irq 12 drq 7 5 on isa0
pcm0. <GUS CS4231> on gusc0
gusc1: <Gravis UltraSound Plug & Play OPL> at port 0x388 0x389 ↵
irq 5 drq 1 on is a0
gusc2: <Gravis UltraSound Plug & P ay MIDI> at port 0x330 0x331 ↵
on isa0
```

На заметку Команда `dmesg` выводит на экран содержимое буфера сообщений ядра. Этот буфер имеет ограниченный размер, и в него постолино поступают новые данные. Таким образом, через какое-то время сообщения, касающиеся идентификации оборудования, будут удалены из буфера.

В сообщениях может указываться несколько драйверов (например, gusc0, gusc1 и gusc2), так как они управляют разными аппаратными компонентами платы. Особую роль играет драйвер pcm0, который выполняет кодирование и декодирование сигналов по алгоритму PCM (Pulse Code Modulation — импульсно-кодовая модуляция). В том или ином виде этот алгоритм применяется в большинстве форматов оцифровки звука.

Если показанные сообщения появляются, значит ядро распознало звуковую плату. Возможно, в модуле звукового драйвера нет необходимости. Если все же нужно загрузить его, воспользуйтесь командой kldload:

```
# kldload snd_gusc.ko
```

Эта команда описывалась в главе 12. В данном случае она загружает драйвер snd_gusc.ko, который поддерживает GUS PnP и родственные звуковые платы.

Некоторые звуковые платы лучше работают, если средства их поддержки скомпилированы в ядре, а не загружаются в виде модулей. Более того, есть драйверы, которые не компилируются как модули и потому должны быть встроены в ядро. При подключении таких плат ядро приходится перекомпилировать.

Независимо от того, встроен ли драйвер в ядро или загружается в виде модуля, FreeBSD требуются файлы устройств для звуковой платы. Во FreeBSD 5.0 эти устройства должны создаваться автоматически файловой системой устройства. В более ранних версиях системы файлы уже существуют в каталоге /dev. Если по какой-то причине их нужно перестроить, введите следующие команды:

```
# cd /dev
# /MAKEDEV snd0
```

Вторая команда создает необходимые файлы устройства. Первым из них следует проверить /dev/sndstat. Отобразите его содержимое и убедитесь в том, что FreeBSD обнаружила звуковую плату:

```
$ cat /dev/sndstat
FreeBSD Audio Driver  newpcm)
Installed devices:
pcm0: <GUS CS4231> at io 0x32c irq  2 drq 7 5 bufsz 4096 %
(1p/1r/0v channels duplex default)
```

Если после строки Installed devices ничего не указано, значит, звуковая плата не найдена. В этом случае просмотрите параметры выбора устройств в конфигурации ядра, как описывалось в главе 12, или список модулей ядра, загруженных для поддержки звука. Возможно, звуковая плата просто неправильно сконфигурирована. В старых платах ISA важные аппаратные характеристики, такие как номер IRQ, часто устанавливались с помощью перемычек. Эти установки не должны конфликтовать с параметрами других устройств, задаваемыми в файле /boot/device.hints:

```
hint.gusc.0.at="isa"
hint.gusc.0.port="0x32c"
hint.gusc.0.irq="12"
```

Эти строки сообщают ядру о том, что устройство gusc подключено к шине ISA через порт 0x32c и номер прерывания — 12.

После выполнения всех вышеперечисленных действий звуковая плата должна быть готова к использованию. Разумеется, к ней необходимо подключить колонки или наушники, чтобы услышать звук. Доступ к звуковой плате осуществляется через устройства типа /dev/dsp0 и /dev/mixer0, но эти детали обрабатываются автоматически соответствующими звуковыми программами.

При использовании наушников начните проверку, установив громкость на минимум, чтобы по неосторожности не быть оглушенным.

Программы работы со звуком

Во FreeBSD поддерживается большое число простейших звуковых программ К ним относятся микшеры, проигрыватели, программы звукозаписи, а также средства работы с файлами и носителями специальных форматов, такими как MP3-файлы и музыкальные компакт-диски. Большинство этих программ доступно как в текстовом так и в графическом вариантах В настоящем разделе будет дан краткий обзор этих программ

На заметку Многие звуковые программы требуют привилегированного доступа к файлам звуковых устройств Во FreeBSD 5.0 файловая система /dev предоставляет всем пользователям право чтения/записи таких файлов Измените права доступа к файлам, если необходимо ограничить круг пользователей, которым разрешено осуществлять запись и воспроизведение звука

Микшеры

Микшеры позволяют регулировать громкость звука на входе и выходе звуковой платы. Простейший микшер называется `mixer` Он инсталлирован по умолчанию и представляет собой утилиту командной строки. В упрощенном виде его синтаксис таков:

```
mixer [ -s] [[устройство] [громкость_слева[:громкость_справа]]]
```

Рассмотрим назначение аргументов

- `-s` При наличии этой опции результаты будут представлены в виде аргументов командой строки утилиты Это позволит сохранить текущие установки с тем чтобы впоследствии их можно было передавать утилите в сценарии запуска
- *устройство* — задает файл звукового устройства Перечень доступных устройств зависит от звуковой платы, но обычно это `vol` (главный регулятор громкости), `pcm` (проигрыватель PCM), `synth` (синтезаторный проигрыватель) `cd` (проигрыватель музыкальных компакт-дисков), `line` (линейный вход) `mic` (микрофон), `igain` (входной усилитель) и `ogain` (выходной усилитель)
- *громкость_слева* и *громкость_справа* Эти опции задают уровни звука в левой и правой колонках в интервале от 0 до 100 где 0 означает отсутствие звука, а 100 — максимальную громкость Если не указать громкость в правой колонке, программа автоматически выставит равный баланс громкости

Существуют и более сложные опции но они используются в системах с несколькими звуковыми платами Описание этих опций доступно на man-странице утилиты

Если не указать никаких аргументов, утилита `mixer` выведет отчет о доступных устройствах и их текущих установках Чтобы изменить ту или иную установку укажите ее название и новое значение:

```
# mixer vol 75
```

Утилиты типа `mixer` полезны в сценариях запуска где с их помощью задаются стандартные уровни громкости Работать с ними в режиме командной строки неудобно хотя некоторые утилиты отображают текстовые меню и регуляторы громкости управляемые с помощью курсорных клавиш Например, на рис. 27.1 представлено окно программы `aumix`, выполняющейся в терминальном окне GNOME.

Есть и графические микшеры отображающие регуляторы громкости различных источников звука. На рис 27 2 изображено окно программы GNOME Mixer. Чтобы

запустить ее, введите **gmix** в командной строке или воспользуитесь экранными меню GNOME. У оболочки KDE тоже имеется свой микшер. Существуют и автономные микшеры, например `xmix`, `xmmix` и `xmixer`. Все они отличаются лишь деталями, в целом работая так же, как и программа GNOME Mixer. Под регуляторами расположены флажки, позволяющие отключать выходные каналы и выбирать входные источники, с которых будет вестись запись звука.

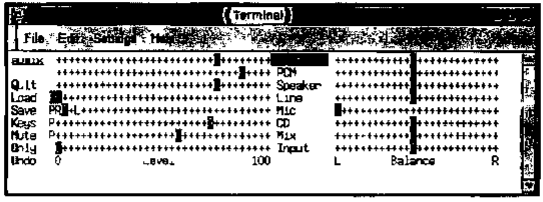


Рис. 27.1. Некоторые микшеры текстового режима отображают визуальные аналоги регуляторов громкости

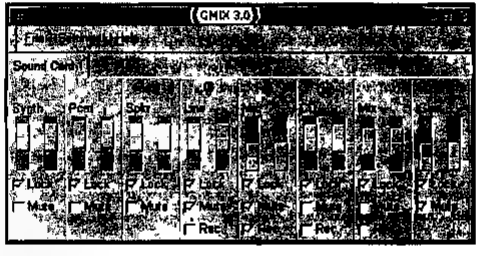


Рис. 27.2. GNOME Mixer и другие графические микшеры отображают регуляторы громкости входных и выходных источников звука

Текстовые утилиты записи и воспроизведения звука

Простейшая утилита записи и воспроизведения звуковых файлов называется `play`. Выполняемые ею функции зависят от способа вызова. Существует символическая ссылка `rec`, включающая режим записи. Программа самостоятельно распознает имя, по которому ее вызывают.

Синтаксис утилиты таков:

```
play [опции] имя_файла [эффект]
```

В простейшем случае достаточно указать имя воспроизводимого файла, например **`play zathras.wav`**. Иногда требуется дать программе определенные подсказки относительно типа файла. Это особенно важно при записи. Перечень доступных опций таков.

- **`-c`** *`каналы`* **или** **`--channels=`***`каналы`* — позволяют задать число звуковых каналов в файле. Стандартные значения — 1 (моно) или 2 (стерео).
- **`-f`** *`формат`* **или** **`--format=`***`формат`* — задают формат записи, т. е. метод кодирования, применяемый при оцифровке данных (не путайте формат записи с ти

пом файла). Возможные форматы таковы: s (линейный со знаком), u (линейный беззнаковый), U (логарифмический U-Law), a (ADPCM), A (логарифмический A-Law) и g (GSM). Все, кроме первых двух, допускают сжатие звука что может приводить к снижению его качества

- **r** *частота* **или --rate=***частота* — задают частоту оцифровки звука в виде количества выборок в секунду. Чем выше частота дискретизации, тем выше качество звука, но и размер файла получается больше. Музыкальные компакт-диски записывают с частотой 44100 Гц, доступной всем современным звуковым платам Для записи голосовых данных достаточно частоты 11250 Гц и даже 8000 Гц, но для записи музыки необходима частота 44100 Гц или даже 48000 Гц
- **s** *размер* **или --size=***размер* — задают размер выборки в битах, т.е. количество битов, кодируемых на каждом шаге оцифровки звуковых данных Возможные варианты таковы: b (8-разрядные целые), w (16-разрядные целые), l (32-разрядные целые), f (32-разрядные с плавающей запятой), d (64-разрядные с плавающей запятой) и D (80-разрядные с плавающей запятой) В большинстве звуковых файлов разрядность выборки составляет 8 или 16 бит
- **-t** *тип* **или --type=***тип* — задают тип записываемого файла. Ему обычно составлено определенное расширение, например .wav для звуковых файлов Windows или au для файлов формата Sun
- **x или --xinu** — позволяют менять порядок байтов в слове для 16- и 32 разрядных форматов. Иногда это необходимо при воспроизведении или подготовке записей для платформ с иным порядком следования байтов.

В качестве примера рассмотрим следующие команды:

```
$ rec -t wav -r 44100 -s w  c 2 sample.wav
$ play sample.wav
```

Первая команда осуществляет запись в файл sample.wav, используя любое из доступных входных устройств в соответствии с текущими установками микшера. Качество записи соответствует компакт диску 44 Кгц 16 бит стерео. По умолчанию команда rec ведет запись бесконечно долго Чтобы остановить ее нажмите <Ctrl+C> Вторая команда воспроизводит записанный файл

Внимание Параметры усиления звука играют чрезвычайно важную роль при записи Если они завышены, самые громкие звуки будут записываться с искажениями, если занижены, звук будет плохо слышно и придется выводить громкость колонок на максимум. Работая с графическими утилитами звукозаписи, можно визуально контролировать уровни громкости

Графические утилиты записи и воспроизведения звука

Утилита play — очень простое средство записи звука. Она мало что сообщает о процессе записи и будучи программой текстового режима, неудобна для пользователей Альтернативой ей служат графические утилиты звукозаписи типа GNOME Sound Recorder (рис. 27.3). Функциональные возможности этой программы в сущности те же, зато у нее графический интерфейс. Для запуска программы введите **grecord** в терминальном окне или воспользуйтесь экранными меню В KDE есть похожая утилита krecord Для записи файла предназначена кнопка Record, для воспроизведения кнопка Play, а чтобы сохранить файл, выберите в меню File команду Save As Окно настроек, вызываемое с помощью команды Settings⇨Preferences, позволяет настраивать такие параметры записи, как частота оцифровки и число каналов. В утилите play/record то же самое делается с помощью опций командной строки

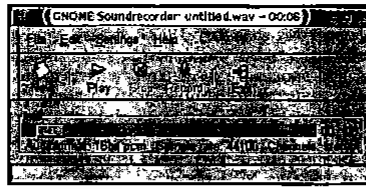


Рис. 27.3. Простейшие графические проигрыватели содержат не так много опций, зато удобны в применении

Более сложные программы типа `glame`, `xwave` и Sweep позволяют редактировать звукозапись, содержащуюся в файле. На рис. 27.4 представлено окно редактирования программы Sweep. В этом окне отображается визуальное представление звукового файла — так называемая звуковая волна. По горизонтальной оси отсчитывается время, по вертикальной — громкость звука. Можно выделять фрагменты звуковой волны и применять к ним различные команды редактирования. К примеру, разрешается уменьшать или увеличивать громкость фрагмента, воспроизводить его в обратном порядке, накладывать на него эхо-эффекты и т.д. Конкретный набор средств редактирования в каждой программе, конечно же, разный.

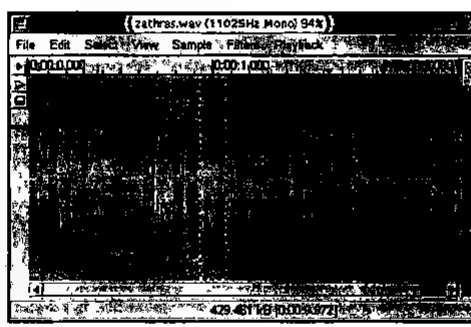


Рис. 27.4. Более сложные звуковые программы позволяют редактировать содержимое звукового файла

Программы для работы с MP3-файлами

Компакт-диски являются популярным средством хранения музыки, но у них есть свои недостатки. Компакт-диск можно потерять или повредить, к тому же при регулярном прослушивании музыки неудобно постоянно менять диски в музыкальном центре. Можно извлечь звуковые дорожки с компакт-диска и сохранить их в виде аудиофайлов, но размеры таких файлов получаются слишком большими. Качество записи компакт-диска — 44 Кгц, 16 бит, стерео. После оцифровки это даст примерно 10 Мбайт данных на каждую минуту звучания. Такие показатели велики даже для современных жестких дисков емкостью 120 Гбайт, ведь на них поместится всего около

200 часов музыки. Для решения этой проблемы был разработан формат сжатия музыки: MP3. Подобно графическому формату JPEG файлы формата MP3 сжимаются с потерей данных. Это делается в целях уменьшения размеров файла. Алгоритм сжатия работает таким образом, что вносимые изменения почти неразличимы для человеческого уха, зато позволяют существенно уменьшить размер файла, сохраняя вполне приемлемое качество звучания. MP3-файлы, записанные с высоким качеством, раз в пять меньше своих несжатых аналогов, а при разумном снижении качества размер получится еще меньше. Вот почему формат MP3 стал настолько популярен у пользователей. Выпущены портативные MP3-проигрыватели, напоминающие по дизайну портативные кассетные или CD-плейеры. Таким образом, важно иметь в наличии программные средства обработки и воспроизведения MP3-файлов.

Большинство музыкальных произведений защищено авторскими правами. Следует учитывать, что копирование музыки в формате MP3 может быть противозаконным действием. Как правило, разрешается копирование сугубо в целях личного использования, для воспроизведения на персональном компьютере или портативном плейере, при условии, что исходный носитель принадлежит вам. А вот передача MP3-записей музыкальных произведений, защищенных авторским правом, своим друзьям — уже незаконна. Это же касается и загрузки MP3-файлов по сети, если только такая загрузка не разрешена владельцем авторских прав на музыкальное произведение (когда вы, скажем, покупаете композицию в MP3 формате на Web-узле правообладателя).

Создание MP3-файлов

Если необходимо создать MP3-файлы на основе имеющегося музыкального компакт-диска, то в первую очередь следует извлечь исходное содержимое компакт-диска в цифровом виде. После получения несжатых аудиофайлов их нужно преобразовать в формат MP3. Ни на одном из этих этапов звуковая плата не задействуется. Иногда оба этапа выполняются одновременно. Например, так "поступает" текстовая утилита `cd2mp3`. Рассмотрим ее применение подробнее.

Для извлечения содержимого компакт-диска необходимо иметь доступ в режиме чтения/записи к дисководу CD-ROM (`/dev/acd0` для дисководов EIDE). Кроме того, утилита `cd2mp3` использует текущий каталог для хранения временных файлов, поэтому ее нужно вызывать только из того каталога, в котором у вас есть право записи, и в той файловой системе, где есть достаточно свободного места для временного хранения несжатого WAV-файла (примерно 10 Мбайт на одну минуту звучания).

После запуска утилита `cd2mp3` задаст ряд вопросов:

1. **Режим кодирования.** Можно выбрать кодирование всего компакт-диска, отдельных дорожек или существующего WAV-файла. Утилита позволяет также извлечь дорожку в WAV-файл, не преобразуя его в MP3-файл.
2. **Частота дискретизации.** Обычно под частотой дискретизации понимают количество выборок в секунду, но применительно к MP3-файлам этот термин приобретает несколько иную трактовку. В данном случае он означает скорость (в килобитах в секунду), с которой файл должен считываться при воспроизведении. Этот показатель важен, если MP3-файл считывается в потоковом режиме, например загружается в реальном времени с Web-сервера. При работе с MP3-файлами на персональном компьютере частоту дискретизации рассматривают как показатель сжатия. Утилита `cd2mp3` поддерживает три значения частоты: 56, 128 и 256. Чем

выше частота оцифровки тем лучше качество звучания и соответственно больше размер файла

3 **Имя выходного файла** Необходимо задать имя выходного MP3-файла Такие файлы имеют расширение `.mp3`

4 **Дисковод CD-ROM**. Утилита `cd2mp3` по умолчанию считывает файлы с устройства `/dev/cdrom`, но оно может отсутствовать в системе поэтому укажите реально присутствующее устройство. Дисководам EIDE соответствует устройство `/dev/acd0`, а дисководам SCSI — `/dev/cd0`. Разрешается также создать символическую ссылку `/dev/cdrom` на реальное устройство

5 **Номер дорожки**. Если требуется извлечь конкретную дорожку, утилита запросит ее номер

Получив необходимую информацию, утилита `cd2mp3` начнет работу. На первом этапе происходит чтение данных с дисковода CD-ROM. Это занимает гораздо больше времени, чем воспроизведение дорожки на проигрывателе музыкальных дисков. Причина заключается в том что музыка на компакт диске хранится в закодированном виде и не так-то просто найти конкретную позицию звуковой дорожки Поэтому, когда утилита `cd2mp3` извлекает аудиоданные, ей приходится сравнивать прочитанный блок с предыдущим блоком, чтобы избежать наложения или пропуска данных Извлеченная дорожка сохраняется в файле `tracknn.wav`, где *nn* — номер дорожки Далее утилита преобразует этот файл в MP3-файл Процесс может занять несколько минут, в зависимости от скорости центрального процессора, поскольку сжатие многомегабайтового WAV-файла — задача, требующая интенсивных вычислений Полученный файл можно проверить с помощью MP3-проигрывателя.

Воспроизведение MP3-файлов

Во FreeBSD есть много MP3 проигрывателей. Укажем некоторые из них

- **`ksmp3play`** текстовый MP3-проигрыватель, использующий библиотеку `curses` для размещения текстовых элементов на экране.
- **`mp3blaster`** — еще один текстовый MP3 проигрыватель основанный на библиотеке `curses`
- **`kmp3`** — MP3-проигрыватель оболочки KDE
- **`replay`** — MP3 проигрыватель оболочки GNOME
- **`x11amp`** — популярный MP3-проигрыватель для X Window System

MP3-файл воспроизводится так же, как и любой другой звуковой файл, хотя у MP3-проигрывателей особый набор опций. Эти программы обрабатывают *списки воспроизведения*, представляющие собой списки файлов, которые следует проигрывать в ходе одного сеанса. Список воспроизведения можно сохранить на диске в виде файла, чтобы впоследствии он загружался одной командой. У некоторых проигрывателей, например `x11amp`, есть графический эквалайзер (рис 27.5).

Проигрыватели музыкальных компакт-дисков

Во многих отношениях проигрыватели компакт-дисков работают так же, как и проигрыватели MP3-файлов, хотя с практической точки зрения есть ряд важных различий

- **Канал передачи звука** Когда дорожка музыкального компакт-диска преобразуется в MP3-файл, то данные считываются с дисковода CD-ROM через его интерфейс EIDE или SCSI, после чего воспроизводятся в виде обычного аудио файла PCM оборудованием звуковой платы Если же компакт-диск воспроиз

водится напрямую, дисковод CD-ROM посылает звуковой плате аналоговые аудиосигналы (иногда оцифрованный звук), которые воспринимаются платой как входные данные от компакт диска. При этом PCM-оборудование не задействуется, что позволяет параллельно слышать звуковые сигналы других программ. Прямое воспроизведение компакт-дисков становится возможным, если дисковод CD-ROM и звуковую плату соединить специальным кабелем.

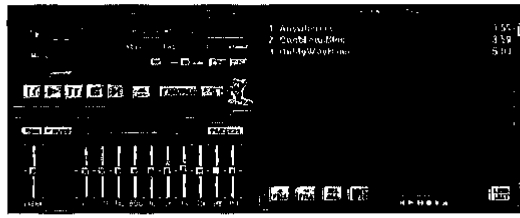


Рис 27 5. Некоторые MP3-проигрыватели имитируют функции традиционных музыкальных центров

- **Наличие носителя** Воспроизводимый компакт-диск должен физически присутствовать в дисководе CD-ROM. В случае проигрывания MP3-файлов, хранящихся на диске, дисковод остается свободным для использования
- **Качество звука** Формат MP3 предусматривает потерю данных, поэтому в сравнении со звуком, воспроизводимым непосредственно с компакт-диска, можно заметить небольшое снижение качества звучания. Впрочем, при частоте оцифровки 256 Кбит и выше это снижение едва ли будет различимо для человеческого уха
- **Составные списки воспроизведения**. При проигрывании компакт дисков невозможно составить список воспроизведения записей нескольких компакт-дисков В случае MP3-файлов в список можно включать файлы, хранящиеся в любых каталогах
- **Нагрузка на центральный процессор** При воспроизведении MP3-файла центральному процессору требуется определенное время на распаковку MP3-данных и передачу их звуковой плате, тогда как проигрывание компакт диска почти не создает нагрузку на процессор

Мультимедийные проигрыватели

До сих пор мы рассматривали программы, работающие лишь со звуковыми данными Но настоящие мультимедийные приложения должны также уметь воспроизводить и/или записывать видеоклипы. Если речь идет только о воспроизведении, то в специальном оборудовании нет необходимости (не считая, конечно, звуковой платы) А, к примеру, в случае видеоконференций требуется подключить к компьютеру цифровую камеру К сожалению, во FreeBSD средства поддержки такого оборудования все еще находятся в зачаточном состоянии, поэтому мы остановимся лишь на программах воспроизведения мультимедийных данных. Обычно они функционируют автономно, но поскольку мультимедийные файлы часто встречаются в Internet, одной из важных задач является подключение проигрывателя к Web-броузеру

Форматы мультимедийных файлов

Мультимедийные файлы бывают разных форматов. Некоторые из них достаточно хорошо поддерживаются во FreeBSD. Ниже перечислены наиболее популярные форматы:

- **AC-3 (ATSC A/52).** Этот метод кодирования применяется в цифровом телевидении и в DVD. Во FreeBSD он поддерживается библиотекой liba52 (`http://liba52.sourceforge.net`).
- **MPEG-1 и MPEG-2.** Организация MPEG (Motion Pictures Experts Group — Экспертная группа по движущимся изображениям) разработала несколько форматов файлов (в том числе, кстати, и упоминавшийся ранее MP3). Форматы MPEG-1 и MPEG-2 широко используются для распространения компьютерных мультимедийных файлов. Поддержка этих форматов во FreeBSD реализована в библиотеке libmpeg2 (`http://libmpeg2.sourceforge.net`) и утилитой MPEG Audio Decoder (`http://www.underbit.com/products/mad/`).
- **RealVideo** — популярный формат, разработанный компанией RealNetworks (`www.real.com`) для сжатия потокового видео. Существует и связанный с ним формат RealAudio той же компании. Для воспроизведения соответствующих файлов необходим проигрыватель RealPlayer. Компания RealNetworks не предлагает этот проигрыватель для FreeBSD, однако Linux-версия проигрывателя работает во FreeBSD при наличии библиотек, обеспечивающих совместимость с Linux. Проигрыватель находится в разделе `audio/linux-realplayer` коллекции переносимых приложений, но в процессе инсталляции нужно обратиться на Web-узел RealNetworks для получения Linux-файла.
- **AVI** — формат компании Microsoft, используемый для распространения мультимедийных файлов в Internet, а также в составе программных пакетов на компакт-дисках. Несмотря на то что такие файлы обычно имеют одинаковое расширение `.avi`, для их создания применяются различные методы кодирования, каждому из которых требуются явные средства поддержки. Многие, но не все AVI-файлы поддерживаются FreeBSD-программой `xanim`.
- **QuickTime** — формат компании Apple, пользующийся умеренной популярностью в Internet. Как несложно догадаться, файлы QuickTime чаще всего встречаются на Web-узлах, предназначенных для пользователей Macintosh. Подобно AVI, формат QuickTime охватывает несколько различных методов кодирования. Некоторые QuickTime-файлы поддерживаются FreeBSD-программой `xanim`.
- **FLIC** — формат компании Autodesk, используемый для создания компьютерной анимации. У него есть два подтипа, различаемых по расширениям `.fli` и `.flc`. Во FreeBSD этот формат поддерживается программой `xanim`.
- **DivX** — разновидность формата MPEG-4, разработанная компанией DivX Networks (`www.divxnetworks.com`). Формат DivX предназначен для распространения видеофильмов по широкополосным соединениям. Ведутся работы по созданию FreeBSD-проигрывателей DivX с открытым исходным кодом, но они еще далеки от завершения. Можно инсталлировать Linux-проигрыватель, требующий наличия библиотек, обеспечивающих совместимость с Linux, он находится в разделе `graphics/divx4linux4` коллекции переносимых приложений.

Во FreeBSD есть мультимедийные библиотеки, предназначенные для поддержки некоторых из перечисленных форматов. Они позволяют программам воспроизводить мультимедийные файлы. Для работы с другими форматами требуются специальные программы, такие как RealPlayer или `xanim`.

Мультимедийные форматы стремительно развивались в последнее десятилетие и наверняка продолжат столь же интенсивно развиваться в будущем. Это приводит к тому, что загруженный файл поддерживаемого типа иногда не удается воспроизвести из-за конфликта версий. В таком случае поищите обновленный проигрыватель или новые версии библиотек поддержки

Одна из важных задач разработчиков FreeBSD — обеспечить возможность воспроизведения видеодисков DVD в системах, оснащенных дисководом DVD-ROM. Существуют различные библиотеки, например liba52, и утилиты типа `xine` (`http://xinehq.de`), которые предназначены для поддержки методов кодирования, применяемых в DVD-дисках. К сожалению, задача усложняется тем, что в большинстве коммерческих DVD-дисков применяется шифрование данных по методу CSS (Content Scrambling System — система засекречивания содержимого). Производители DVD-проигрывателей и разработчики программ воспроизведения DVD-дисков для Windows и MacOS могут приобрести у организации DVD Copy Control Association (DVD-CCA) соответствующие ключи дешифрования, однако эта организация не предлагает такие ключи для проектов с открытым исходным кодом, поэтому в настоящее время во FreeBSD отсутствуют коммерческие DVD-приложения. Неудивительно, что алгоритм шифрования CSS был взломан и появились библиотеки функций для декодирования зашифрованных DVD-дисков, подключаемые к `xine` и другим программам FreeBSD. Впрочем, по закону Digital Millenium Copyright Act (DMCA) распространение таких библиотек в США запрещено. В 2002 г. рассматривалось несколько связанных с этим судебных дел, и есть надежда, что в будущем во FreeBSD появятся легальные средства воспроизведения DVD-дисков.

Алгоритм CSS применяется в коммерческих видеодисках, но не для шифрования компьютерных данных на DVD-дисках. Таким образом, дисковод DVD-ROM можно использовать во FreeBSD для чтения информационных DVD-дисков. Даже сама система распространяется на [illegible] носителях.

Воспроизведение мультимедийных файлов

Для воспроизведения мультимедийных файлов необходимы соответствующие проигрыватели. Рассмотрим наиболее популярные программы:

- **`xanim`** (`http://xanim.polter.net`). Эта программа уже давно является стандартным мультимедийным проигрывателем во FreeBSD и других UNIX-системах. Она поддерживает большое число форматов, включая многие разновидности AVI и QuickTime.
- **`xine`** (`http://xinehq.de`). Эта программа стремительно завоевывает популярность. В ней сделан акцент на поддержке форматов высококачественного видео, включая форматы DVD-дисков. Выпускаются подключаемые модули, расширяющие возможности программы, в том числе модули CSS-дешифрования (официально доступны только за пределами США).
- **RealPlayer**. Этот проигрыватель воспроизводит файлы RealAudio и RealVideo, встречающиеся на многих Web-узлах. Программа доступна в коллекции переносных приложений, но предварительно нужно загрузить Linux-файл с Web-узла компании RealNetworks (`www.realnetworks.com`).

Для воспроизведения видеоклипа нужно ввести имя проигрывателя, а за ним имя искомого файла, например:

```
$ xanim kinetoscope.avi
```

Появится окно для отображения клипа Некоторые программы выводят в этом ок не элементы управления, но они могут находиться и в отдельном окне как в случае программы xanim (рис. 27.6)

Рис 27 6. Проигрыватели видеоклипов воспроизво дят видеоизображение вместе со звуковой дорожкой

Подключение проигрывателя к броузеру

В Internet можно найти множество мультимедийных и звуковых файлов Web-броузеры позволяют закреплять за соответствующими типами файлов программы-обработчики для воспроизведения файлов непосредственно в процессе загрузки Для этого нужно связать с проигрывателем определенный тип MIME (Multipurpose Internet Mail Extensions — многоцелевые расширения электронной почты в сети Internet). Детали этой процедуры зависят от конкретного броузера, но общая схема остается одинаковой. В качестве примера возьмем броузер Mozilla.

1 Запустите броузер, введя `mozilla` в терминальном окне или выбрав соответст вующую ссылку в экранных меню

2 Выберите в меню Edit команду Preferences Появится диалоговое окно Preferences с установками броузера

3 Выберите в левой панели пункт Navigator | Helper Applications чтобы отобразить окно MIME-настроек (рис 27 7)

4 Щелкните на кнопке New Type Программа отобразит диалоговое окно New Type, в котором нужно задать четыре параметра описание типа (для справки), расширение файлов (такое как `.wav` или `.avi`), тип MIME и полный путь к программе обработчику В табл 27.1 указаны типы MIME для некоторых наиболее распространенных мультимедийных типов файлов со ссылками на соответствующие программы.

5 Щелкните на кнопке OK в диалоговом окне New Type, после чего описание соз данного типа MIME должно появиться в окне Preferences

6. Щелкните на кнопке OK в окне Preferences

Если вы не знаете, какой тип MIME соответствует нужному типу фай ла, и в системе инсталлирован сервер Apache, просмотрите файл `/usr/local/etc/apache2/mime.types` Аналогичные списки доступ ны и в Internet, например по адресу `http://www december com/html/spec/mime.html`.

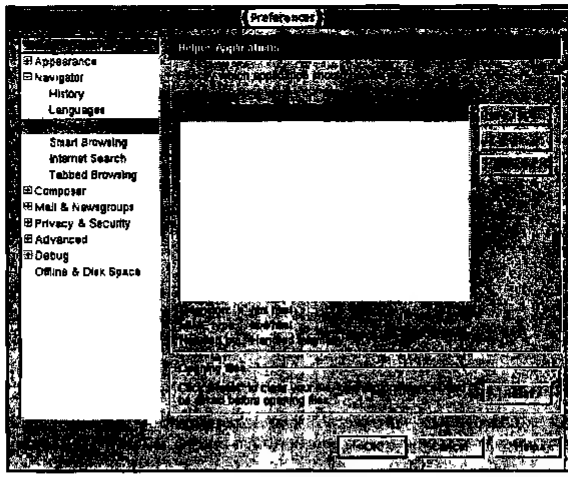


Рис 27 7. В разделе Helper Applications окна Preferences броузера Mozilla можно устанавливать привязки между приложениями и типами MIME

Таблица 27.1. Распространенные мультимедийные типы файлов их расширения соответствующие типы MIME и возможные проигрыватели во FreeBSD

Тип файла	*Расширение*	*Тип MIME*	*Возможный обработчик*
Звуковые WAV файлы	.wav	audio/x-wav	/usr/local/bin/play
Звуковые MP3-файлы	.mp3	audio/mpeg	/usr/local/bin/xllamp
Видеоклипы MPEG	mpg mpeg mpe	video/mpeg	/usr/X11R6/bin/xanim
Видеоклипы QuickTime	mov qt	video/quicktime	/usr/X11R6/bin/xanim
Видеоклипы AVI	avi	video/x-msvideo	/usr/X11R6/bin/xanim
Файлы RealAudio	ra	audio/x realaudio	/usr/local/bin/realplay
Файлы RealAudio	rpm	audio/x pn realaudio plugin	/usr/local/bin/realplay
Файлы RealAudio	ram	audio/x-pn-realaudio	/usr/local/bin/realplay
Файлы RealVideo	.rv	video/vnd.rn-realvideo	usr/local/bin/realplay
Файлы RealMed a	rm	application/vnd rn-realmedia	/usr/local/bin/realplay

Если за MIME-типом закреплена программа обработчик то при щелчке на соответствующей гиперссылке Web-броузер запустит эту программу Для запуска Linux-программ необходимо наличие библиотек бинарной совместимости с Linux

Внимание В некоторых случаях, например когда файл содержит программный код, автоматический запуск обработчика нежелателен. Иногда возможна ошибка в интерпретации типа. В частности, расширение `.rpm` имеют не только файлы проигрывателя RealPlayer, но и файлы пакетов RPM (Red Hat Package Manager), поэтому Web-серверы могут неправильно распознавать MIME-типы

Игры

Большинство игр являются мультимедийными приложениями, хотя встречаются и исключения: во FreeBSD есть много старых текстовых игр, которые не заслуживают того, чтобы называться "мультимедийными". Игры, доступные во FreeBSD, довольно просты по современным стандартам. К сожалению, популярные коммерческие игры выпускаются только для Windows, изредка для Mac OS. Есть игры для Linux, и многие из них могут выполняться во FreeBSD при наличии библиотек, обеспечивающих совместимость с Linux. Можно даже запускать некоторые Windows-игры под управлением пакета WINE (см. главу 4).

Текстовые игры UNIX

В 70-х и 80-х годах прошлого века, когда операционная система UNIX переживала эпоху юности, компьютерные игры были совершенно примитивны, по современным представлениям. Они функционировали в текстовом режиме и могли использовать лишь текст и псевдографику, например вертикальные линии и дефисы.

Многие игры такого рода находятся в разделе `games` коллекции пакетов или коллекции переносимых приложений, хотя и не выделены в отдельную группу. Следует сказать пару слов об одном классе текстовых игр, которые дожили до сегодняшних дней и даже продолжают развиваться: это *приключенческие игры*, получившие название от классической игры Adventure. В них моделируются сюжеты, подобно приключенческому роману, излагаемые в текстовом виде. Главная роль в сюжете отводится пользователю, который должен вводить инструкции, определяющие сюжетные ходы. Ниже приведен небольшой фрагмент игрового сеанса из коммерческой игры *Planetfall* за пущенной в интерпретаторе Frotz.

```
Escape Pod
This is one of the Feinstein's primary escape pods, for use in
extreme emergencies. A mass of safety webbing, large enough to
hold several dozen people, fills half the pod. The controls are
entirely automated. The bulkhead leading out is open.

Спасательный модуль
Это один из основных спасательных модулей "Файнстайна",
предусмотренный на случай аварии. Множество ложементов, в расчете
на несколько десятков человек, заполняет половину модуля.
Управление полностью автоматизировано. Наружный люк открыт

The ship shakes again. You hear  from close by  the sounds of
emergency bulkheads closing

Корабль снова трясет. Вы слышите, как закрываются аварийные люки

>wait
Time passes

Время летит
```

```
Through the viewport of the pod you see the Feinstein dwindle as
you head away. Bursts of light dot its hull. Suddenly, a huge
explosion blows the Feinstein into tiny pieces, sending the escape
pod tumbling away!

Сквозь иллюминатор модуля вы видите, как "Файнстайн" удаляется
от вас, постепенно сокращаясь в размерах. Вспышки света очерчивают
его корпус. Внезапно огромный взрыв разносит корабль на куски,
и спасательный модуль летит кувырком!

You are thrown against the bulkhead, head first. It seems that
getting in the safety webbing would have been a good idea.

Вас швыряет о стенку люка головой вперед. Умные люди всегда пристегиваются

   **** You have died ****
        Вы умерли
```

В приключенческих играх используются простые команды на английском языке как **wait** ("подожди") в предыдущем примере. Часто распознается и более сложный синтаксис, например **insert the golden coin into the vending machine** ("бросьте монетку в автомат"). Играя в приключенческую игру, будьте готовы "умереть" несколько раз, прежде чем вам удастся успешно завершить сюжет.

Большинство современных приключенческих игр написано на языке, известном как *z-код*. Он использовался в популярных играх 80-х годов компании Infocom. Старые и новые игры, разрабатываемые энтузиастами, можно запускать с помощью любого доступного интерпретатора z-кода, например Frotz, Nitfol, `xinfocom`, `zip` (не имеет отношения к одноименной утилите архивирования файлов) и Jzip. Крупный архив приключенческих игр доступен по адресу `http://www.ifarchive.org/indexes/if-archiveXgamesXzcode.html`. Там собраны преимущественно бесплатные игры, написанные разработчиками со всего мира.

Еще один класс текстовых игр, доживших до нашего времени, основан на древней UNIX-игре Rogue. В ней игроки соревнуются в прохождении лабиринта, по ходу дела собирая различные сокровища и сражаясь с монстрами. От нее ведут начало такие игры, как NetHack, Angband и Moria. Некоторые из потомков имеют графический интерфейс. Вообще говоря, многие современные игры, например Diablo и Quake, позаимствовали немало идей у Rogue.

Игры для X Window System

Современные графические игры для UNIX используют либо X Window System, либо библиотеку SVGAlib. Спектр игр достаточно широк: от простых карточных до весьма сложных "стрелялок" и "стратегий". Как и в случае текстовых игр, многие из них доступны в разделе `games` коллекции пакетов или коллекции переносимых приложений.

В оболочках GNOME и KDE есть коллекции игр, представленные пакетами `gnomepackages` и `kdegames` соответственно. После инсталляции они станут доступными через экранные меню оболочки. В основном это довольно простые программы: карточные игры, головоломки и т.д. Встречаются и гораздо более сложные игры. Рассмотрим некоторые из них.

- **GNU Chess**. Ядро программы GNU Chess является не слишком сложным, но существует несколько графических надстроек к нему, например GNOME Chess. Эти надстройки предлагают пользователям более понятный интерфейс.
- **FreeCiv**. Стратегическая игра, похожая на коммерческую игру Civilization, но в отличие от оригинала являющаяся сетевой. Она состоит из серверного и кли

ентского пакетов. Чтобы играть в нее на одном компьютере, инсталлируйте и запустите оба пакета. Сервер вычисляет параметры игрового пространства, а клиент управляет взаимодействием с пользователем. Такая схема позволяет легко переходить в многопользовательский режим. Пользователь другого компьютера может установить у себя только клиентскую часть и подключиться к серверу.

- **LinCity.** Аналог коммерческой игры SimCity. Программа моделирует город, которым нужно управлять, строя виртуальные здания, распределяя городской бюджет и т.д.

Бессмысленно рассказывать о том, как играть в эти игры, так как у них разные интерфейсы, задачи и т.д. В играх типа GNU Chess или FreeCiv, предусматривающих очередность ходов, есть опции меню или объекты в игровом окне, позволяющие вносить изменения в игру. Иногда, чтобы сделать ход, нужно явно выбрать определенную опцию. Существуют также игры реального времени, в которых действия пользователей почти мгновенно отражаются на экране, например различные "стрелялки" типа `xasteroids`.

Коммерческие игры

Для FreeBSD выпускается не так много коммерческих игр, поскольку рынок FreeBSD слишком мал для окупаемости таких игр. Более привлекательной платформой является Linux. Для этой операционной системы выпущен ряд популярных игр, включая Quake III Arena, Civilization: Call to Power и SimCity 3000. К сожалению, перенесом многих коммерческих игр в Linux занималась компания Loki, которая прекратила существование, поэтому их теперь сложно найти. Кроме того, нет гарантий, что та или иная игра будет работать во FreeBSD при наличии библиотек, обеспечивающих совместимость с Linux.

Во FreeBSD можно запускать некоторые игры Windows, используя пакет WINE, который описывался в главе 4. Это, кстати, была одна из целей разработки пакета, поэтому игры функционируют в среде WINE даже лучше, чем некоторые офисные приложения. Тем не менее пакет WINE далеко не совершенен, и его не так-то просто конфигурировать. Прежде чем покупать коммерческую игру, проверьте ее в базе данных WINE Application Database (`http://appdb.codeweavers.com`). Полезным информационным ресурсом является также телеконференция `comp.emulators.mswindows.wine`.

Резюме

Мультимедийные приложения — не самая сильная сторона FreeBSD, поскольку разработчики системы традиционно направляли свои усилия на оптимизацию стека сетевых протоколов, написание серверов и даже создание бизнес-приложений. Тем не менее в распоряжении пользователей FreeBSD есть целый ряд мультимедийных программ — от простых утилит командной строки, предназначенных для записи и воспроизведения аудиофайлов, до сложных программ редактирования звука и проигрывателей видеоклипов. Во FreeBSD имеются также игры, хоть они и довольно просты по современным стандартам. В то же время в системе можно запускать некоторые игры, написанные для других платформ, в частности для Linux и Windows.

Полный справочник по

Часть VI

Обслуживание системы

Полный справочник по

Глава 28

Автоматизированные и неавтоматизированные процедуры обслуживания

Большая часть книги посвящена конфигурированию FreeBSD или ее повседневному применению. Но в этой главе, которой начинается заключительная часть книги, мы поговорим о том, как обеспечивать корректную работу системы. Первый вопрос, который следует рассмотреть, — регламентное обслуживание, т.е. процедуры, выполняемые на регулярной основе, например ежедневно или еженедельно. Мы сделаем акцент на четырех задачах обслуживания: удаление ненужных файлов, контроль загруженности центрального процессора, контроль использования памяти и обновление программного обеспечения системы. Эффективное решение этих задач позволяет избежать многих проблем, таких как ошибки переполнения диска, приводящие к потере файлов, или низкая производительность, вызванная неразумной эксплуатацией ресурсов процессора либо памяти. Лучше устранять проблемы на ранних стадиях, чем провоцировать жалобы пользователей.

Очистка файлов

Компьютеры чем-то напоминают наши квартиры: в них тоже скапливается хлам. Но если в квартире хламом являются старые газеты, кипы ненужных бумаг, детские игрушки, какие-то тряпки и т.д., то роль компьютерного "хлама" играют старые файлы. Это могут быть устаревшие журнальные файлы, временные файлы, которые по какой-то причине не были удалены, неиспользуемые программные файлы и разного рода пользовательские файлы. Точно так же, как после уборки мусора в квартире становится легче дышать, удаление ненужных файлов упрощает жизнь системному администратору.

Журнальные файлы

В некоторых случаях очень важно иметь возможность проследить действия той или иной программы, чтобы выявить источник проблемы, проверить факт взлома системы или просто убедиться в том, что система работает корректно. Для этих целей многие программы (особенно серверы) фиксируют свои действия в *журнальных файлах*. Большинство журнальных файлов FreeBSD находится в каталоге `/var/log`. Среди наиболее важных файлов упомянем `messages` (хранит общие системные сообщения), `security` (хранит сообщения, касающиеся безопасности системы), `auth.log` (хранит аутентификационную информацию), `sendmail.st` и `maillog` (хранят информацию о действиях почтового сервера `sendmail`; первый из них является бинарным), а также `cron` (фиксирует действия планировщика `cron`). Остальные файлы содержат сведения о конкретных серверах или подсистемах. Внимательно просмотрите содержимое каталога `/var/log`, чтобы узнать, какие серверы работают в системе. В некоторых главах книги мы ссылались на конкретные журнальные файлы, в основном как на источник отладочной информации.

Журнальные файлы накапливают информацию, поэтому их размер увеличивается. Теоретически это означает, что такие файлы могут со временем занять все свободное дисковое пространство. Чтобы избежать этой неприятности, необходимо периодически очищать журнальные файлы. Чаще всего выполняется *ротация*, при которой активный журнальный файл объявляется устаревшим и переименовывается, возможно с последующим сжатием в целях экономии места на диске, а затем создается и открывается новый файл с таким же именем. Например, файл `/var/log/messages` после переименования и сжатия будет называться `/var/log/messages.0.gz`, а система продолжит записывать сообщения в новый файл `/var/log/messages`. На следующем цикле ротации файл `messages.0.gz` будет назван `messages.1.gz`, файл `messages`

опять переименуется в messages.0.gz, после чего будет создан новый файл messages. На определенном этапе система удалит файл messages.#.gz, вместо того чтобы переименовывать его.

Ротацию журнальных файлов выполняет утилита newsyslog, которая вызывается планировщиком cron (описывался в главе 6). Конфигурационный файл этой утилиты называется /etc/newsyslog.conf. Он содержит строки комментариев, начинающиеся с символа '#', и по одной строке для каждого контролируемого журнального файла. Формат записей таков:

```
журнальный_файл [владелец:группа] режим счетчик размер когда
[флаги] [файл_идентификатора] [номер_сигнала]
```

Рассмотрим назначение полей записи.

- *журнальный_файл*. В первом поле указан полный путь к файлу.
- [*владелец:группа*] Второе поле — необязательное. В нем задаются владелец и группа, которым принадлежат журнальные файлы, подвергаемые ротации, например root:wheel.
- *режим* Необходимо задать режим доступа к архивируемым журнальным файлам, например 644 (кто угодно может читать файлы) или 600 (файлы может читать только владелец).
- *счетчик* В этом поле задается число, определяющее, сколько журнальных файлов участвует в цепочке архивирования.
- *размер*. Здесь задается пороговый размер в килобайтах, по достижении которого выполняется архивирование исходного журнального файла. Например, если это поле равно 100, утилита newsyslog заархивирует файл, как только его размер превысит 100 Кбайт. Если указать специальный символ '*' утилита будет игнорировать размер файла, ориентируясь на следующее поле записи.
- *когда* По умолчанию утилита newsyslog вызывается системным планировщиком cron каждый час и проверяет поле *размер*, чтобы определить, нужно ли архивировать журнальный файл. В такой конфигурации поле *когда* содержит символ '*'. С другой стороны, можно выполнять принудительную ротацию в заданное время, указывая символ '*' в поле *размер* и код даты — в поле *когда*. Если поле начинается с символа '@', значение времени интерпретируется в формате ISO 8601. Если же в начале поля стоит символ '$', то можно использовать специальные символы D, W и M для указания интервалов времени в течение дня, недели и месяца. Например, выражение $M1D6 означает, что ротация выполняется в 6:00 утра в первый день каждого месяца, а выражение $W7D22 означает ротацию в 10:00 вечера (D22) каждую субботу (W7). Подробнее эти форматы описаны на man-странице утилита newsyslog.
- [*флаги*] Это необязательное поле служит двум целям. Во-первых, с его помощью можно заставить утилиту newsyslog сжимать архивы, используя утилиту gzip (флаг Z) или bzip2 (флаг J). Во-вторых, если журнальный файл имеет бинарный формат, то в этом поле нужно указать флаг B. Обычно утилита добавляет к файлу текстовое сообщение о произошедшей ротации, но в случае бинарных файлов такое сообщение нежелательно.
- [*файл_идентификатора*]. Некоторым серверам нужно сообщать о том, что произошла ротация журнальных файлов. Чтобы отправить такое уведомление, утилита newsyslog должна знать идентификатор серверного процесса (PID). Многие серверы хранят свои идентификаторы в файлах, имена которых задаются в этом необязательном поле. Чтобы узнать имя файла, обратитесь к документации на сервер.

- [*номер_сигнала*]. Если указан файл, содержащий идентификатор процесса, утилита `newsyslog` посылает серверу определенный сигнал. По умолчанию это сигнал SIGHUP, в ответ на который большинство серверов повторно читает свой конфигурационный файл и открывает новый журнальный файл, что позволяет утилите `newsyslog` заархивировать старый. Но некоторые серверы не корректно реагируют на сигнал SIGHUP; в этом случае следует указать другой сигнал в данном поле. За необходимой информацией обращайтесь к документации на сервер.

Как правило, менять конфигурацию журнальных файлов не приходится, но если какой-то файл начинает сильно разрастаться, проверьте установки файла `/etc/newsyslog.conf`, чтобы узнать, осуществляется ли ротация. Если нет, добавьте соответствующую инструкцию и посмотрите, как пройдет ротация. Другие изменения диктуются потребностями конкретной системы.

Временные файлы

Многим программам приходится создавать временные файлы. В них могут содержаться обрабатываемые данные, резервные копии данных на случай сбоя программы, данные для передачи другой программе и т. д. Теоретически каждый временный файл должен быть удален создавшим его процессом или последующей программой в цепочке обработки, но на практике это происходит не всегда. На нормальный ход событий влияют разные факторы: программные ошибки, сбои программы или системы (возможно, вследствие внешних причин, таких как выключение питания). В результате в системе остаются "осиротевшие" временные файлы. Со временем они накапливаются и начинают занимать слишком много дискового пространства.

В качестве одной из мер борьбы с указанными проблемами FreeBSD выделяет отдельные каталоги специально для временных нужд. Наиболее важными из них являются `/tmp`, а также `/var/tmp` и `/usr/tmp`. Именно там создаются временные файлы большинства программ. Отсюда вытекают три важных следствия:

- Временные каталоги можно размещать в специально предназначенных для этого разделах. Такой подход, рассмотренный в главе 7, означает, что повреждения, вызываемые программой, которая создает слишком большие временные файлы, локализуются во временных разделах. Это не мешает пользователям сохранять файлы в своих начальных каталогах, например.
- Системному администратору достаточно просматривать лишь несколько каталогов в поисках слишком старых или чрезмерно больших временных файлов.
- Можно написать сценарий для поиска и автоматической обработки временных файлов.

Упомянутый сценарий создается по умолчанию при инсталляции системы. Существует задание планировщика `cron`, организующее вызов утилиты `periodic` для выполнения регламентного обслуживания. Она, в свою очередь, вызывает сценарий `/etc/periodic/daily/100.clean-tmps` для удаления старых временных файлов из каталога `/tmp`. Этот сценарий избавляет систему от ненужного "хлама", но он не лишен недостатков. В частности, каталоги `/var/tmp` и `/usr/tmp` приходится проверять самостоятельно. Кроме того, некоторые программы создают временные файлы в нестандартных каталогах. Например, программа GIMP (см. главу 26) задействует для этого начальный каталог пользователя. На с. 552 мы поговорим о том, как следить за пользовательскими файлами, в том числе временными, находящимися в начальных каталогах.

Что делать, если обнаружены старые или чрезмерно большие временные файлы? Во первых, проверьте дату их создания. Относительно недавние файлы (скажем, не

более чем дневной давности) могут еще использоваться; их не стоит удалять в ходе регламентного обслуживания если только они не вызывают серьезных проблем. Файлы, созданные до того, как произошла загрузка системы (время загрузки определяется командой **uptime**), однозначно являются устаревшими и могут быть безопасно удалены Если есть сомнения, определите владельца файла (по команде **ls -l**) и спросите у него, нужны ли ему эти файлы Файлы, владелец которых не зарегистрирован в системе (определяется командой **who**), скорее всего не используются, но это не факт возможно, они созданы планировщиками `cron` или `at`

Неиспользуемые программы

Неиспользуемые программы занимают место на диске и рано или поздно этот фактор сыграет свою роль. Более того, такие программы потенциально являются уязвимым местом с точки зрения безопасности Если в одной из них будет выявлена серьезная ошибка, она станет мишенью для злоумышленника. Риск особенно возрастает, если программа принадлежит пользователю `root` и у нее установлен бит SUID.

Просмотрите основные каталоги программ (`/usr/local/bin`, `/usr/local/sbin` и `/usr/X11R6/bin`) и выясните статус каждой программы. Можно воспользоваться командой `pkg_info`, чтобы узнать, к какому пакету принадлежит файл, например

```
$ pkg_info -W /usr/X11R6/bin/xwave
/usr/X11R6/bin/xwave was installed by package xwave 2 2
$ pkg_info xwave-2.2
Information for xwave 2 2

Comment
Audio player/recorder/editor for the X W ndow System
```

На заметку Некоторые программные файлы, особенно в каталогах `/bin`, `/sbin` `/usr/bin` и `/usr/sbin`, не представлены в пакетной базе данных FreeBSD. Эти программы были инсталлированы либо как базовая часть системы, либо путем компиляции исходного кода в обход коллекции переносимых приложений

Вторая команда `pkg_info` в предыдущем примере возвращает дополнительную информацию, на основании которой можно решить, нужна ли данная программа Если она, что называется, "отработала свое", удалите ее с помощью команды `pkg_delete`, как описывалось в главе 11

```
# pkg_delete xwave-2.2
```

Указанные действия не обязательно выполнять регулярно. Просто, разобравшись как следует в работе системы, просмотрите еще раз список инсталлированных в ней программ и определите, какие из них вполне можно удалить

Учет пользовательских файлов

Пользователи часто захватывают под свои нужды слишком много дискового пространства. Здесь на помощь приходит команда `du`, которая отображает объем дискового пространства, занимаемого указанным каталогом и каждым из его подкаталогов. Чтобы узнать, сколько места на диске занимает конкретный каталог со всеми подкаталогами, добавьте опцию `-s`, например

```
# du -s /home/sskinner
61034   /home/sskinner
```

Команда сообщает о том, что начальный каталог пользователя `sskinner` содержит 61034 блоков. В большинстве FreeBSD-систем размер блока составляет 1 Кбайт, по

этому приведенное значение равно 61034 Кбайт Используйте команду `du` для контроля дискового пространства

Внимание Если пользователям разрешено хранить файлы в общих каталогах, одной команды `du` будет недостаточно, хотя она позволяет определить размеры таких общих каталогов

В небольших системах команду `du` применять удобно, однако в крупных системах оценка начального каталога каждого пользователя превращается в утомительное занятие. Для таких случаев во FreeBSD предусмотрен механизм *квот*, позволяющий системе самостоятельно следить за использованием дискового пространства и блокировать операции создания файлов в случае превышения лимита Включать механизм квот имеет смысл лишь в многопользовательских системах Рассмотрим, как это делается

1. **Скомпилируйте поддержку квот в ядре** Стандартное ядро FreeBSD не содержит средств поддержки квот. Перекомпилируйте ядро, как описывалось в главе 12, только добавьте строку `options QUOTA` в файл конфигурации ядра
2. **Разрешите поддержку квот на этапе начальной загрузки** Добавьте следующие строки в файл `/etc/rc conf` чтобы заставить систему использовать механизм квот

```
enable_quotas="YES"
check_quotas="NO"
```

На заметку Параметр `check_quotas` сообщает FreeBSD о том, нужно ли проверять согласованность квот на этапе начальной загрузки Эта проверка может занимать много времени, поэтому ее обычно отключают.

3. **Смонтируйте дисковые разделы с поддержкой квот** В файле `/etc/fstab` нужно добавить параметр `userquota` для каждой файловой системы, которая монтируется с поддержкой пользовательских квот, и параметр `groupquota` для каждой файловой системы, в которой должны использоваться групповые квоты Например, строка для файловой системы `/home` может выглядеть так:

```
dev/ad4s3h     home    ufs    rw,userquota,groupquota  2  2
```

4. **Создайте базы данных квот** Введите **`quotacheck -a`**, чтобы создать низкоуровневые базы данных с информацией о квотах в соответствующих файловых системах.
5. **Включите режим квот** Введите **`quotaon -a`**, чтобы система начала вести учет квот во всех файловых системах, где это разрешено
6. **Отредактируйте параметры квот для обычного пользователя.** Введите **`edquota -u`** *`пользователь`*, чтобы изменить квоты указанного пользователя, или **`edquota -g`** *`группа`*, чтобы изменить квоты заданной группы. FreeBSD запустит редактор определяемый значением переменной среды `EDITOR`, и загрузит в него временный файл с установками пользовательских квот. В этом файле можно задать *жесткий лимит* (`hard`), определяющий предельно допустимый размер дискового пространства, и *нежесткий лимит* (`soft`), определяющий максимальный размер дискового пространства на длительный период времени Оба значения выражаются в блоках, которые обычно равны 1024 байт Например, если жесткий лимит равен `100000`, а нежесткий `75000`, то пользователь может постоянно хранить 75000 файловых блоков, но на короткое время превышать это значение вплоть до 100000 блоков Продолжительность этого "короткого времени", определяется в исходном коде ядра Можно также воспользоваться опцией `-t` команды `edquota`, чтобы задать соответствующее значение самостоятельно Разрешается также менять число *индексных дескрипторов*, задействуемых пользователем Индексный де

скриптор — это служебная структура данных, связанная с файлом. Таким образом, лимит индексных дескрипторов определяет максимальное число файлов создаваемых пользователем. Редактируемый файл квот выглядит примерно так:

```
Quotas for user sskinner:
/usr: blocks in use: 61034, limits (soft = 75000  hard = 100000
        inodes in use: 6391, limits (soft = 7500  hard = 10000)
```

7. **Продублируйте установки обычного пользователя.** Воспользуйтесь опцией `-p` команды `edquota`, чтобы продублировать значения квот для группы пользователей. Например, команда **`edquota -p sskinner 1001-9999`** распространяет значения квот пользователя `sskinner` на всех пользователей с идентификаторами от 1001 до 9999.

8. **Измените значения квот для особых случаев.** Если каким-то пользователям требуется назначить иные значения квот, повторите этап 6 (и, возможно, 7).

Желательно протестировать функционирование механизма квот. Создайте тестовую учетную запись и попытайтесь превысить для нее лимит квот (допустим, путем копирования большого числа крупных файлов). Если ограничения окажутся не такими, как ожидалось, проверьте, все ли установки сделаны правильно.

Контроль загруженности центрального процессора

Производительность центрального процессора - понятие неоднозначное. Иногда мощность процессора избыточна для системы. Например, типичная рабочая станция, применяемая для обработки текста и решения других задач, узким местом которых является время реакции пользователя, не нуждается в вычислительных мощностях, характерных для типичного современного компьютера. Если же система функционирует в многопользовательской среде либо применяется для выполнения очень трудоемких расчетов, таких как трассировка лучей в программах трехмерной анимации, инженерное моделирование или анализ научных данных, то мощности процессора часто оказывается недостаточно. В этих случаях (особенно в многопользовательских системах) приходится тщательно следить за загруженностью центрального процессора. Если одновременно запустить много трудоемких заданий, работа компьютера может неприемлемо замедлиться. Такие проблемы иногда возникают и в однопользовательских системах, когда какая-нибудь программа "зависает", но при этом продолжает обращаться к процессору, хотя вроде бы ничего не делает (даже ее окно возможно уже закрыто). Системный администратор должен уметь определять загруженность процессора и устранять связанные с этим проблемы.

Утилиты для контроля процессора

Одно из основных средств контроля процессора — команда `uptime`. Она сообщает о том, сколько времени прошло с момента загрузки системы, сколько пользователей зарегистрировано в системе и каковы средние показатели загруженности процессора за последнюю минуту, пять минут и пятнадцать минут. Рассмотрим пример.

```
# uptime
10:53AM  up 1 day  20:17, 2 users  load averages: 0.06  0.02  0.01
```

В этом примере средняя загруженность за последнюю минуту составляет 0,06, за последние пять минут — 0,02 и за последние пятнадцать минут — 0,01. Показатель 1,00 означает, что объем запрашиваемых ресурсов в точности соответствует объему

доступных ресурсов. Если запустить ресурсоемкую программу на однопользователь ской рабочей станции при отсутствии других ресурсоемких программ, то показатель средней загруженности составит как раз примерно 1,00. В случае двух таких программ показатель превысит 1,00. Запросы программ не будут удовлетворяться в полной мере, и их производительность ухудшится. Таким образом, оптимальная средняя загруженность составляет 1,00

На заметку

Как правило, обращения к процессору происходят всплесками", когда программы отображают окна, выполняют вычисления и т.д. Затем загруженность снижается почти до нуля, поскольку программы переходят в режим ожидания или завершают работу. Таким образом, даже если средняя загруженность невысока, система может работать недостаточно быстро из-за того, что "всплески" длятся дольше, чем это было бы в случае более производительного процессора

Если команда `uptime` сообщает о высокой средней загруженности процессора, необходимо выяснить, что является тому причиной Для этого существует команда `top`, отображающая список программ, упорядоченный по интенсивности использования процессора Список автоматически обновляется каждые несколько секунд, поэтому можно в реальном времени наблюдать за обращениями к процессору. На рис 28 1 приведены результаты работы команды `top`. В колонках `WCPU` и `CPU` выводятся два показателя, касающиеся использования процессора, причем сортировка выполняется по столбцу `CPU` В колонке `command` указано имя команды, обращающейся к процессору. Как видите, около половины ресурсов захватила программа `xllamp`, еще 10% приходится на долю программы `Xvnc` Влияние остальных программ минимально

На заметку

Показатели загруженности, сообщаемые командой `top`, нельзя напрямую сравнивать с показателями команды `uptime`, т.к. они никогда не превышают 100%. Команда `top` учитывает реальное выделение ресурсов, а команда `uptime` — лишь запросы к ресурсам, поэтому ее показатели могут превышать 1 0

Следует учитывать и другие колонки отчета команды `top` Наиболее важная из них — `PID`, где указаны идентификаторы процессов Они нужны для уничтожения или изменения приоритетов процессов, потребляющих слишком много ресурсов В колонке `USERNAME` указывается владелец процесса что тоже немаловажно

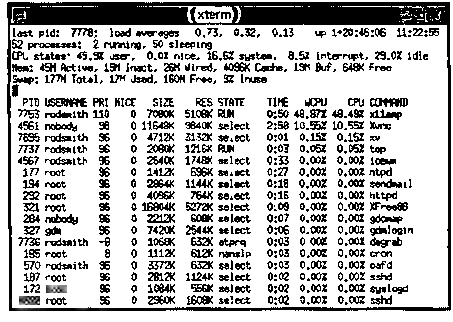


Рис. 28.1 Команда `top` позволяет выявлять программы потребляющие слишком много ресурсов процессора

Выявление случаев чрезмерной загруженности процессора

Не существует простого правила позволяющего определить, когда загруженность процессора является чрезмерной для данного компьютера Конечно, программы будут демонстрировать неоптимальную производительность при показателе средней загруженности выше 1,0, однако это значение нельзя считать решающим критерием. Если FreeBSD-компьютер играет роль сервера дистанционной регистрации для множества пользователей (подключающихся через SSH, X Window System или VNC, как описывалось в главе 21), то, скорее всего, средняя загруженность окажется выше 1,0, поскольку пользователи будут часто запускать программы одновременно. Даже на одно пользовательской рабочей станции может одновременно работать несколько ресурсоемких приложений, что приводит к значительному повышению средней загруженности. Подобная ситуация может встречаться на многопользовательском компьютере, сервере или интенсивно эксплуатируемой рабочей станции и не обязательно означает наличие проблем.

Чрезмерной загруженностью следует считать ситуацию, при которой система демонстрирует неприемлемую производительность. Естественно, проблемы производительности возникают не только из-за процессора, но и вследствие недостаточного объема оперативной памяти, как объясняется в следующем разделе. Если система работает слишком медленно, а показатель средней загруженности значительно превышает 1,0 (равен, к примеру, 5,0), обратите пристальное внимание именно на использование процессора.

Если команда `top` сообщает о том, что ресурсы процессора захвачены странными процессами или программами, о наличии которых вы и не подозревали, выясните, что они собой представляют и устраните проблему Можно также предпринять определенные действия для того чтобы система оставалась доступной даже при высоких показателях средней загруженности

Устранение проблем

Перегрузки могут возникать из-за *зависших* процессов. Это процессы, которые перестали реагировать на действия пользователя. Иногда они не причиняют особого вреда, а лишь занимают память. В других случаях процессы "зависают" из за того, что они заблокировали сами себя в бесконечном цикле Такие процессы потребляют ресурсы процессора и приводят к тому, что показатель средней загруженности становится равным 1,0 (при отсутствии других процессов). Их легко выявить с помощью команды `top`, поскольку они будут расположены в начале списка. Понятно что не всякий процесс, потребляющий много ресурсов, является зависшим Прежде чем уничтожать процесс, узнайте, кому он принадлежит и что делает этот пользователь.

Для уничтожения зависших процессов предназначена команда `kill` (см. главу 5) Предположим, нужно избавиться от процесса с идентификатором 7297 Для этого следует ввести такую команду

```
# kill 7297
```

Проверьте, удалось ли избавиться от процесса Большинство зависших процессов не реагирует на обычную команду `kill`, поэтому нужно применить более жесткий подход и послать процессу сигнал `KILL` (номер 9), который означает принудительное завершение программы самой системой

```
# kill -KILL 7297
```

Иногда процесс является вполне легитимным, но потребляет много ресурсов процессора Это приводит к снижению производительности, что отражается на работе пользователей Например, программы распределенных вычислений, такие как

SETI@Home (http://setiathome.ssl.berkeley.edu), очень ресурсоемки и выполняются длительное время, но нежелательно, чтобы это влияло на обработку текста, просмотр Web-страниц и т.д. В этом случае нужно воспользоваться командой nice, чтобы запустить программу с меньшим приоритетом, или, если программа уже работает, выполнить команду renice и понизить ее приоритет. Обе эти команды описывались в главе 5. Как средство устранения проблем команда renice полезнее, поскольку проблема не всегда очевидна в момент запуска программы. Предположим, к примеру, что в небольшой многопользовательской системе нормальный показатель средней загруженности составляет 1–2, но может вырасти до 3 из-за того, что какой-то пользователь запустил ресурсоемкую программу. Определив с помощью команды top, что идентификатор нужного процесса составляет 7297, воспользуемся следующей командой для уменьшения приоритета программы

```
# renice 10 7297
```

На заметку С помощью команды killall можно уничтожить процесс по имени, а не по номеру. Это избавит вас от необходимости выяснять идентификатор процесса. Но у медали есть обратная сторона: команда killall удаляет *все* процессы с заданным именем независимо от того, который из них "завис", а который — нет.

Первый аргумент команды, 10, — это новый приоритет процесса. У большинства процессов FreeBSD приоритет 0. Чем выше значение приоритета, тем ниже приоритетность процесса. Наименее приоритетные процессы имеют приоритет 20. Таким образом, значение 10 является хорошей отправной точкой для исправления ситуации.

Недостатком такого подхода является то, что ресурсоемкий процесс позже закончит свою работу, ведь он теперь будет реже получать доступ к центральному процессору. Это особенно справедливо для систем со средней загруженностью 2,0. В системах с незначительной загруженностью (менее 1,0) снижение приоритета ресурсоемкого процесса мало повлияет на продолжительность его работы, поскольку он все равно будет получать доступ к процессору в периоды простоя других программ, что судя по средней загруженности, происходит довольно часто.

Предотвращение проблем

Есть несколько путей устранения проблем, связанных с загруженностью центрального процессора.

- **Приобретение более мощного оборудования.** Если в системе работает много пользователей или регулярно выполняются ресурсоемкие программы, оснастите ее достаточно производительным процессором. К сожалению, определить точный показатель производительности довольно сложно. Выбирая материнскую плату, обратите внимание, чтобы она поддерживала процессоры с гораздо более высокой частотой, чем у приобретаемого процессора. Впоследствии это позволит обновить сам процессор, не меняя материнскую плату.
- **Запуск более эффективных программ.** Важную роль играет эффективность самих программ. Компиляция программы с учетом особенностей конкретного процессора позволит получить выигрыш в производительности, но еще больший выигрыш даст замена одной программы другой, в которой используются более эффективные алгоритмы.
- **Запуск меньшего числа программ.** Чем больше программ работает в системе, тем меньше ее производительность. Например, на рис. 28.1 видно, что программа xlamp (MP3-проигрыватель, рассматривавшийся в главе 27) захватила почти

половину ресурсов процессора Если завершить ее освободится много ресурсов для более важных программ.

- **Использование команды nice** Ресурсоемкие программы можно запускать с помощью команды `nice`, чтобы уменьшить их влияние на интерактивные программы Например, команда `nice` *программа* запускает указанную программу с приоритетом 10
- **Регулярная проверка загруженности процессора**. Работая в многопользовательской системе, регулярно проверяйте показатели загруженности процессора. Если наблюдается неожиданное увеличение средней загруженности, проверьте не является ли это следствием "зависания" какого-то процесса. Сложно помешать процессу "зависнуть", но зато, если быстро его уничтожить пользователи не успеют ощутить снижение производительности

Контроль использования памяти

FreeBSD скрывает детали распределения памяти от рядовых пользователей и их программ Каждой программе выделяется собственная область памяти, отделенная от областей памяти других программ. Всей выделенной памятью управляет ядро системы При запуске программ, которые не помещаются в физической памяти, ядро задействует *область подкачки* (пространство на диске, используемое для расширения физической памяти) Естественно, работа с областью подкачки ведется медленнее чем с физической памятью, и если потребность в памяти слишком велика, то даже область подкачки может исчерпаться Это означает, что при повышении числа запросов к памяти производительность системы начнет снижаться, а после определенного момента программы откажутся запускаться Вот почему так важно контролировать использование памяти во FreeBSD

Утилиты для контроля памяти

Чтобы иметь возможность контролировать использование памяти, нужно знать две важные вещи. сколько памяти есть в системе и сколько памяти необходимо программам. На основании этого уже можно оценить, достаточно ли памяти, и предпринять какие то действия для устранения проблем, если они возникли

Определение доступной памяти

Чтобы выяснить объем имеющейся памяти воспользуйтесь командой `sysctl`

```
$ sysctl hw.physmem
hw.physmem: 96870400
$ sysctl hw.usermem
hw.usermem  73068544
```

Переменная `hw.physmem` равна общему объему физической памяти, установленной в компьютере, за исключением небольшого объема, зарезервированного для BIOS, аппаратных устройств и т.д. Переменная `hw.usermem` равна общему объему памяти, доступному для пользовательских программ Это значение всегда меньше, чем `hw.pnysmem`, поскольку часть памяти ядро резервирует для себя Оба значения сообщаются в байтах, так что в данном примере на компьютере установлено 92 Мбайт оперативной памяти из которых 70 Мбайт находится в распоряжении пользовательских программ Эти значения не учитывают область подкачки Ее параметры можно узнать с помощью команды `top`

Определение потребности в памяти

Упоминавшаяся ранее команда `top` сообщает об использовании не только процессора, но и памяти. Взгляните еще раз на рис. 28.1 и посмотрите на четвертую и пятую строки. В них приводятся показатели использования памяти и области подкачки. В строке `Mem:` сообщается о том, сколько памяти (включая физическую и область подкачки) выделено. Строку `Swap:` интерпретировать проще. Рассмотрим ее отдельно:

```
Swap: 177M Total, 17M Use   160M Free, 9% Inuse
```

Здесь говорится о том, что из 177 Мбайт области подкачки используется 17 Мбайт. Первый показатель является суммарным по всем разделам подкачки в системе. Это разделы были созданы в ходе инсталляции FreeBSD.

Строка `Mem:` сложнее для анализа, и по ней тяжело судить: достаточно ли памяти в системе. FreeBSD динамически выделяет память различным процессам по мере необходимости, включая память для кэширования операций доступа к диску. В связи с этим сообщается, что объем свободной памяти мал даже в системе с невысокой степенью загруженности. Лучше полагаться на суммарные значения, сообщаемые командой `sysctl`. В рассматриваемом примере компьютер имеет 92 Мбайт ОЗУ, из которых 70 Мбайт доступно для пользовательских программ.

Команды `top` и `ps` сообщают объемы памяти, занимаемые отдельными программами. В колонках `SIZE` и `RES` на рис. 28.1 просуммирован общий размер выделенных страниц памяти и размер резидентных страниц. Команда `ps au` выдает список всех процессов, выполняемых на компьютере, а также различную статистическую информацию о них. В колонках `VSZ` и `RSS` будут отображаться объемы занимаемой памяти. Все эти сведения можно использовать для выявления программ, запрашивающих слишком много памяти.

Выявление случаев чрезмерной загруженности памяти

Как и в случае центрального процессора, ответ на вопрос о том, когда загруженность памяти становится чрезмерной, таков: когда начинают возникать проблемы. Это может быть замедление работы системы, вынужденной часто обращаться к области подкачки, или невозможность запуска программы либо выполнения ею операций из-за нехватки памяти.

Как правило, если задействуемый объем области подкачки начинает приближаться к общему объему оперативной памяти, проблемы производительности неизбежны. В рассматривавшемся выше примере, где объем оперативной памяти составлял 92 Мбайт, 17 Мбайт используемой области подкачки не означали нехватку памяти, так как второе значение намного меньше, чем первое, а значит, система тратит не слишком много времени на операции страничного обмена. Вот если бы используемый объем области подкачки регулярно достигал 90 Мбайт, можно было бы сделать вывод о том, что система нуждается в дополнительном объеме оперативной памяти.

Другого рода проблемы возникают, когда используемый объем области подкачки приближается к общему ее объему. Если бы в рассматриваемом примере размер области подкачки составлял 25 Мбайт, то используемая доля в 17 Мбайт означала бы ситуацию, близкую к критической. В такой системе лучше расширить область подкачки, как будет описано на с. 560. Снизить потребность в обращениях к области подкачки можно также, добавив оперативную память.

На заметку

Традиционный совет заключается в том, что размер области подкачки должен в два раза превышать объем физической памяти. Предполагается, что при нормальной загруженности системы объем используемой памяти равен объему ОЗУ, поэтому у компьютера будет хороший буфер на случай внезапного всплеска обращений к памяти.

Замеряйте показатели использования памяти тогда, когда это наиболее актуально. Нетрудно догадаться, что в 5 утра в воскресенье загруженность системы будет низкой, а вот останется ли она такой в полдень рабочего дня?

Если есть подозрение что какая то программа занимает слишком много памяти проверьте ее несколько раз с помощью команды `ps -au`. При постоянном возрастании используемых объемов памяти может иметь место *утечка памяти* Это такая ситуация, когда программа запрашивает память, а потом "забывает" ее освободить. "Утечки" редко случаются в конечных версиях программ, чаще — в бета-версиях Чем больше "утечка", тем серьезнее возникающая проблема. Единственный способ ее устранения — остановить и перезапустить программу

Предотвращение и устранение проблем

Существует три возможности устранить проблемы, связанные с памятью

- **Наращивание ОЗУ** Если система испытывает нехватку памяти, будет наблюдаться снижение производительности и программы начнут давать сбои Современные компьютеры поддерживают сотни мегабайтов и даже тигабайты оперативной памяти, причем цена на память снизилась настолько, что добавление памяти часто является наиболее простым выходом из ситуации.
- **Увеличение области подкачки.** Недостаточный размер области подкачки вызывает проблемы в периоды особо интенсивного использования памяти О том, как решить эту проблему, будет рассказано ниже
- **Сокращение потребности в памяти.** Если нельзя ни добавить оперативную память, ни расширить область подкачки, то некоторые проблемы можно решить снизив общую потребность в памяти. Один из вариантов инсталлировать менее ресурсоемкую графическую оболочку вместо KDE или GNOME (о таких оболочках говорилось в главе 23) Много ресурсов потребляют броузер Mozilla и пакет OpenOffice.org На сервере можно выгрузить X Window System

Увеличение области подкачки

Если в ходе инсталляции FreeBSD не была создана область подкачки достаточного размера или если потребность в памяти увеличилась, расширьте область подкачки. Это можно сделать путем создания дополнительного раздела подкачки или файла подкачки

Добавление раздела подкачки

Если жесткий диск уже разбит на разделы и конфигурация системы зафиксирована, то добавление нового раздела является утомительной процедурой, поскольку требуется создать резервные копии одного или несколько разделов, удалить эти разделы потом воссоздать их с новыми размерами и восстановить их данные Процедуры управления разделами подробно рассматривались в главе 7. Данные можно архивировать на ленту или жесткий диск с помощью команды `tar`, но не исключено, что для восстановления данных потребуется резервная система. Вообще, лучше не менять структуру разделов рабочей системы

Раздел подкачки удобно создавать при добавлении в систему нового жесткого диска Запись о новом разделе следует поместить в файл `/etc/fstab` По окончании воспользуйтесь командой `top`, чтобы проверить распознала ли система новый раздел

Совет Поскольку скорость страничного обмена существенно сказывается на общей производительности системы, располагайте раздел подкачки так чтобы доступ к нему осуществлялся максимально быстро Лучший

вариант — посреди физического диска (между другими разделами). Это позволит свести к минимуму время, необходимое для перемещения головки диска

Добавление файла подкачки

В отсутствие нового жесткого диска область подкачки лучше всего расширить с помощью специального файла, интерпретируемого системой как раздел подкачки. Процедура создания такого файла различается во FreeBSD 4.*x* и 5.0. Работа с файлом ведется не так эффективно, как с разделом подкачки, поскольку этот файл может быть фрагментирован, что вызовет дополнительное перемещение головки диска. Тем не менее этот незначительный недостаток с лихвой компенсируется простотой создания файла подкачки. Во FreeBSD 5.0 процедура настолько проста, что в экстренных случаях, например при запуске ресурсоемкой программы, можно расширить область подкачки на разовой основе.

FreeBSD 4.x

Чтобы создать файл подкачки, выполните перечисленные ниже действия.

1. Перекомпилируйте ядро, как описывалось в главе 12, включив строку для драйвера vn в файл конфигурации ядра.

```
pseudo-device   vn   1   # драйвер виртуальных файлов
```

2. Создайте устройство для виртуальных файлов с помощью следующих команд.

```
# cd /dev
# sh MAKEDEV vn0
```

3. Создайте с помощью команды `dd` пустой файл нужного размера, который станет файлом подкачки. Например, следующая команда создает файл размером 100 Мбайт в корневом разделе.

```
# dd if=/dev/zero of=/swap0 bs=1048576 count=100
```

4. Задайте права доступа к файлу равными 0600.

```
# chmod 0600 /swap0
```

5. Добавьте следующую строку в файл `/etc/rc.conf`, чтобы файл подкачки подключался на этапе начальной загрузки системы.

```
swapfile="/swap0"
```

6. Перезагрузите систему, чтобы подключить ядро, скомпилированное в п. 1, и файл подкачки, созданный в пп. 2–5.

FreeBSD 5.0

Во FreeBSD 5.0 прежний драйвер виртуальных файлов заменен драйвером резидентного диска (`md`). Этот драйвер включен в стандартное ядро, поэтому его не нужно перекомпилировать. Чтобы включить поддержку файла подкачки, выполните указанные ниже действия.

1. Создайте с помощью команды `dd` пустой файл нужного размера. Например, следующая команда создает файл размером 100 Мбайт в корневом разделе.

```
# dd if=/dev/zero of=/swap0 bs=1048576 count=100
```

2. Воспользуйтесь утилитой `mdconfig`, чтобы заставить систему сформировать интерфейс устройства для созданного файла. Следующая команда возвращает имя устройства (скорее всего `md0`, означающее, что оно доступно по ссылке `/dev/md0`).

```
# mdconfig -a -t vnode -f /swap0
```

3 Задайте права доступа к файлу равными 0600

```
# chmod 0600 /swap0
```

4 Введите **swapon /dev/md0**, чтобы подключить область подкачки (измените аргумент /dev/md0, если утилита mdconfig сообщила другое имя устройства)

5. Добавьте следующую строку в файл /etc/rc.conf, чтобы файл подкачки автоматически подключался на этапе начальной загрузки системы:

```
swapfile="/swap0"
```

Обновление программного обеспечения

Один из аспектов регламентного обслуживания, требующий постоянного внимания, — это обновление программного обеспечения. Здесь задействуются многие процедуры, описанные в главе 11, где речь шла об управлении пакетами и переносимыми приложениями

Важность обновлений

Зачем обновлять программу после того как она была инсталлирована? Для этого есть три основные причины, две из которых тесно связаны между собой

- **Улучшение функциональных возможностей** Новые версии программ содержат новые функции, которые делают программу более простой в использовании более производительной или в чем-то другом превосходящей предыдущую версию. Естественно, новые функции могут и не понадобиться пользователям, но не мешает узнать, какие возможности добавлены в постоянно используемую программу.
- **Исправление программных ошибок** Часто обновления нацелены на исправление ошибок в программном коде. Эти ошибки могут приводить к некорректной работе программы, недопустимым погрешностям при вычислениях и сбоям программы. Следовательно, такого рода обновления важны, если программа используется для решения актуальных задач
- **Устранение "брешей"** Некоторые обновления связаны с устранением проблем, касающихся безопасности пользователей или системы. Например, серверная программа может содержать ошибку, позволяющую получать несанкционированный доступ к компьютеру, возможно даже в режиме суперпользователя, извне. Когда появляется такое обновление, *обязательно* установите его при первой же возможности

От некоторых функциональных улучшений можно отказаться если они не играют особой роли. Даже некоторые программные ошибки нет смысла исправлять, если они проявляются в ситуациях, не имеющих непосредственного отношения к вам, например на другой аппаратной платформе или в иной программной среде

Средстаа и процедуры проверки обновпвний

Существует три метода обновления системы. Первый из них заключается в применении стандартных утилит для обновления отдельных пакетов или переносимых приложений по мере необходимости. Такой подход требует, чтобы пользоваталь или администратор отслеживал появление обновлений для всех используемых программ или по крайней мере для тех из них, которые могут нуждаться в обновлениях Второй ме

тод состоит в применении команды `make world`, которая перекомпилирует все программы на компьютере с использованием самых последних версий исходных кодов Это очень трудоемкий подход, но его стоит применять, если система порядком устарела Последний метод заключается в переинсталляции FreeBSD. Это наименее приемлемый подход, ведь он означает, что придется все конфигурировать заново Зато появится возможность "начать с чистого листа", если конфигурация системы стала слишком запутанной и не поддается улучшению

Какие программы обновлять

Иногда сложно определить, какие именно пакеты требуют обновления Описываемая ниже утилита `portupgrade` способна составить перечень всех пакетов и переносимых приложений, которые можно обновить. Команда `make world` обновляет вообще все, что есть в системе. Конечно, это не самый оптимальный вариант. Лучше придерживаться методики избирательных обновлений, но тогда нужно самостоятельно решать нуждается ли та или иная программа в обновлении Наиболее важные обновления связаны с устранением "брешей", поэтому следует быть в курсе событий в мире компьютерной безопасности Рекомендации по вопросам безопасности FreeBSD публикуются по адресу `http://www.freebsd.org/security`. Есть также специализированные Web-узлы и списки рассылки, поддерживаемые, в частности, организациями CIAC (Computer Incident Advisory Capability — консультативная служба по компьютерным сбоям; `http://www.ciac.org/ciac/`) и CERT/CC (Computer Emergency Response Team Coordination Center координационный центр группы компьютерной "скорой помощи", `www.cert.org`) Если пакеты были получены из нестандартных источников, регулярно проверяйте эти источники, чтобы вовремя узнать о важных обновлениях.

Обновления не связанные с безопасностью менее критичны Просто периодически проверяйте их появление и выясняйте, какие функциональные возможности предлагаются в новых версиях

Пакетные обновления

Утилита `portupgrade` является чрезвычайно популярным средством обновления программ на уровне отдельных пакетов Если она не инсталлирована в системе, поищите ее в разделе `sysutils` коллекции пакетов или коллекции переносимых приложений Утилита поддерживает большое число опций, поэтому не забудьте просмотреть ее man-страницу В упрощенном виде синтаксис утилиты таков:

```
portupgrade [опция] [название_пакета]
```

Рассмотрим назначение основных опций

- **`-a`, `--all`** — заставляет утилиту обновить все пакеты, инсталлированные в системе.
- **`-f`, `--force`** — заставляет утилиту заменить пакет, даже если новая" версия имеет тот же (либо меньший) номер, что и у инсталлированного пакета Такой режим используется, если в процессе обновления экспериментального пакета возникли проблемы и нужно вернуться к последней стабильной версии
- **`-g`, `--go-on`** Обычно, если пакет зависит от других пакетов утилита `portupgrade` требует, чтобы зависимые пакеты были заранее инсталлированы или успешно обновлены Данная опция заставляет утилиту игнорировать отсутствие необходимых пакетов Это удобно, если они инсталлируются в обход пакетной базы данных FreeBSD
- **`-i`, `--interactive`** При наличии этой опции утилита запрашивает подтверждение перед инсталляцией или обновлением каждого пакта

- **-n --noexecute** — заставляет утилиту выполнить "пробный проход", т.е. сообщить о том, какие пакеты будут инсталлированы, но не выполнять собственно инсталляцию. Утилита в любом случае загрузит немалый объем данных, что займет почти столько же времени, как и при полном обновлении.
- **-p, --package** — заставляет утилиту создать пакет для каждого компилируемого переносимого приложения. Это может понадобиться, если обновляется несколько систем. В таком случае готовые пакеты можно скопировать из первой системы во все остальные, сэкономив время на компиляции переносимых приложений в каждой системе.
- **-P --use-packages** Обычно утилита `portupgrade` проверяет не только набор пакетов, но и коллекцию переносимых приложений. Эта опция заставляет утилиту всегда, когда это возможно, работать с пакетами в целях экономии времени (к сожалению, утилите иногда не удается корректно найти хранилище пакета). Учтите, что пакеты не всегда бывают оптимизированы для конкретной системы.
- **-PP, --use-packages-only**. Эта опция аналогична предыдущей, но *запрещает* утилите использовать коллекцию переносимых приложений. Если пакет недоступен, никакого обновления не произойдет.
- **-r, --recursive** — вызывает обновление всех пакетов, зависящих от указанного.
- **-y --yes** В процессе инсталляции некоторых пакетов и переносимых приложений выдаются разного рода запросы. Эта опция заставляет утилиту отвечать на все такие запросы утвердительно.
- **-R, --upward-recursive** — вызывает обновление всех пакетов, от которых зависит указанный пакет.
- *название пакета*. Можно работать с отдельным пакетом (например, `samba`) или с группой пакетов, указывая метасимволы в кавычках (такие как `"*"` или `"?"`) или расширенные регулярные выражения, идентифицируемые начальным двоеточием.

Прежде чем запускать утилиту `portupgrade`, вызовите утилиту `pkgdb`, чтобы проверить, находится ли пакетная база данных в согласованном состоянии. Соответствующая команда выглядит так:

```
# /usr/local/sbin/pkgdb -F
```

Она может выполняться несколько минут, особенно в системе с большим числом инсталлированных пакетов или медленным процессором либо контроллером жесткого диска. При обнаружении каких-либо несогласованностей утилита выдаст запрос по поводу дальнейших действий. В большинстве случаев нужно отвечать **yes**, так как это позволит устранить проблему. Иногда, правда, приходится действовать наобум. Например, утилита может спросить, нужно ли заменить одну зависимость другой, но не понятно, оправданна ли такая замена. Впрочем, ни один из ответов не повлияет на работу имеющихся программ, поскольку изменения отразятся лишь на последующей инсталляции пакетов.

Убедившись в целостности пакетной базы данных, выполните команду `portupgrade` с опциями `-n` и `-P` и выражением `"*"` вместо имени пакета, чтобы узнать, какие программы требуют обновления. Команда выдаст серию сообщений, касающихся пакетных обновлений. Чаще всего встречаются сообщения `No need to upgrade`, но некоторые строки более содержательны. Анализ результатов работы команды `portupgrade` отнимает много времени, поэтому лучше перенаправить их в файл:

```
# /usr/local/sbin/portupgrade -nP "*" 2> upgrade.txt
```

Даже в режиме простой проверки утилита `portupgrade` выполняется достаточно долго, ведь ей все равно приходится загружать обновления. Если вы решились обновить все, опустите опцию `-n`, а если нужно обновить лишь некоторые программы, укажите их имена в командной строке:

```
# /usr/local/sbin/portupgrade -P samba gplot apache pine
```

Эта команда обновляет четыре пакета `samba gplot apache` и `pine`.

Обновления в стиле make world

Пользователи FreeBSD часто предпочитают компилировать программы из исходных кодов. Это позволяет создавать исполняемые файлы, оптимизированные для данного компьютера. Некоторые программисты считают, что так надежнее, чем доверять другим. Во FreeBSD предусмотрено специальное средство для перекомпиляции всех программ на основе имеющихся исходных кодов, команда `make world`. Она позволяет обновлять программы до уровня не только последних стабильных версий, но и последних версий, находящихся в стадии разработки. Таким образом, после выполнения команды `make world` система может стать менее стабильной.

На заметку

Несмотря на то что мы говорим о команде `make world`, она сама считается устаревшей. Вместо нее нужно вводить последовательность аналогичных команд, как будет описано ниже.

Широкомасштабное обновление системы не сводится к выполнению одной лишь команды типа `make world`. Это довольно утомительная процедура, не говоря уже о том, насколько она опасна. Сбой на каком-нибудь этапе может привести к тому, что система станет непригодной к работе. Необходимо также перезагружать компьютер, что нежелательно для промышленных систем, которые должны постоянно работать. Наконец, это очень длительный процесс, который может занимать несколько часов даже на весьма быстродействующем компьютере.

Внимание

Не выполняйте команду типа `make world` просто ради интереса! Приводимые ниже инструкции намеренно являются не столь подробными, как обычно. Если вы не можете самостоятельно разобраться в деталях описываемой процедуры, то скорее разрушите систему, вместо того чтобы обновить ее. Более подробная информация приведена в руководстве *FreeBSD Handbook* по адресу `http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook`.

Сделав необходимое предупреждение, рассмотрим базовую процедуру.

1. **Разберитесь в назначении описываемой процедуры.** Прежде всего подпишитесь на список рассылки FreeBSD-STABLE или FreeBSD-CURRENT, в зависимости от обновляемой версии системы (стабильной считается последняя версия обычного выпуска системы, а текущей — экспериментальная версия предварительного выпуска). Ознакомьтесь с содержимым списка рассылки в течение нескольких дней, чтобы понять, с какими проблемами можно столкнуться. Инструкции, касающиеся правил подписки, приведены по адресу `http://www.freebsd.org/support.html#mailing list4`.
2. **Инсталлируйте исходные коды системы.** Инсталлируйте с помощью программы `sysinstall` исходные коды всей системы. Для этого перейдите в меню `Configure⇨Distributions⇨src` и выберите `All`.
3. **Обновите исходные коды системы.** Воспользуйтесь командой `cvsup`, описанной в главах 11 и 12, для обновления исходных кодов системы. Конфигурационный

файл /etc/cvsupfile должен содержать строку src all означающую обновление всех исходных кодов

4 **Создайте резервную копию системы** В связи с потенциальным риском настоятельно рекомендуется создать резервную копию системы, прежде чем продолжать процедуру Создайте также копии каталогов /etc и /usr/local/etc, чтобы в экстренных случаях можно было восстановить рабочие конфигурационные файлы

5 **Прочтите файл /usr/src/UPDATING** Этот файл содержит информацию о версии компилируемого программного обеспечения, включая сведения о потенциальных опасностях и известных ошибках

6 **Проверьте и модифицируйте файлы /etc/defaults/make conf и /etc/make conf**. Эти файлы содержат стандартные команды, используемые для компиляции программ Измените установки по умолчанию с учетом особенностей системы Можно, к примеру, добавить параметры оптимизации для конкретного процессора Если содержимое файлов непонятно, лучше не трогайте их чтобы не повлиять негативным образом на все последующие действия

7 **Обновите файлы /etc/group и /etc/passwd** После выполнения команды cvsup будет получено два набора конфигурационных файлов: один в каталоге /etc, другой — в каталоге /usr/src/etc Последний включает все изменения, необходимые для обновленных программ Одно из них касается новых групп и пользователей в системных файлах паролей. Если обновить систему, не добавив соответствующие учетные записи, она может некорректно работать или вообще откажется загружаться. Сравните файлы /etc/group и /etc/passwd с файлами /usr/src/etc/group и /usr/src/etc/passwd и добавьте необходимые учетные записи

8 **Перейдите в однопользовательский режим** Введите **shutdown now** на системной консоли (от имени суперпользователя), чтобы перейти в однопользовательский режим. Будут остановлены работающие серверы и выведены из системы все пользователи что позволит безопасно выполнить ее обновление

9 **Введите указанные четыре команды** Следующие команды позволяют убедиться в том, что все файловые системы смонтированы правильно и область подкачки используется:

```
# fsck -p
# mount -u /
# mount -a -t  ufs
# swapon -a
```

10 **Установите системные часы**. Если аппаратные часы настроены на локальное время, введите **adjkerntz -i**, чтобы система установила корректное время Если этого не сделать, при компиляции некоторых файлов могут возникнуть странные ошибки

11 **Перейдите в каталог исходных кодов** Введите **cd /usr/src** чтобы перейти в основной каталог исходных кодов

12 **Скомпилируйте приложения** Введите **make buildworld**, чтобы скомпилировать все программы Этот процесс может занять несколько часов

13. **Скомпилируйте ядро**. Введите **make buildkernel**, чтобы скомпилировать ядро В главе 12 описывалось, как настроить ядро для конкретной системы Это важный этап, так как иначе будет компилироваться ядро generic

14 **Инсталлируйте приложения** Введите **make installworld** чтобы инсталлировать исполняемые файлы скомпилированные в п. 12

15 **Инсталлируйте ядро** Введите `make installkernel` чтобы инсталлировать ядро скомпилированное в п 13.

16 **Проверьте изменение конфигурационных файлов** Воспользуйтесь утилитой `mergemaster` для сравнения файлов каталога `/etc` с новыми стандартными файлами в каталоге `/usr/src/etc` Если ввести **`mergemaster`**, будет создан отчет о найденных отличиях. Каждое отличие представлено в формате утилиты `diff`. Не которые отличия отражают изменения, сделанные пользователем. Они имеют право на существование Другие изменения внесены разработчиками FreeBSD. Постарайтесь отследить их и реализовать самостоятельно путем копирования из мененного файла или редактирования существующего.

17 **Перезагрузите систему** Введите **`fastboot`** чтобы загрузить новую систему

Если некоторые из перечисленных действий кажутся вам непонятными, *не выполняйте обновление.* В команде типа `make world` редко возникает необходимость Это скорее средство для опытных администраторов FreeBSD, которые хотят повысить быстродействие компьютера или по-особому его настроить. Рекомендуется опробовать процедуру на тестовой системе, прежде чем браться за рабочий компьютер Выясните какие проблемы чаще всего возникают и как от них избавиться, прежде чем рисковать важными данными

Переинсталляция FreeBSD

Еще более радикальной процедурой обновления является переинсталляция FreeBSD. Она необходима, если система слишком устарела и не отвечает требованиям времени. Возможно также, что конфигурация системы серьезным образом нарушилась и необходимо начать все сначала.

Об инсталляции FreeBSD подробно рассказывалось в главе 2, поэтому не будем повторяться. Посоветуем лишь создать резервные копии каталогов `/etc` и `/usr/local/etc`, так как они позволяет воссоздать конфигурацию которая существовала на момент первой инсталляции.

Чтобы не уничтожать пользовательские файлы, не удаляйте первичный раздел FreeBSD или тот раздел, в котором эти файлы существуют. Настоятельно рекомендуется создавать резервные копии пользовательских файлов в качестве страховки на случай неудачной инсталляции

Резюме

FreeBSD содержит средства позволяющие выполнять автоматизированные регламентные процедуры, в частности ротацию журнальных файлов и очистку каталогов временных файлов. Другие аспекты регламентного обслуживания требуют участия администратора В частности, необходимо следить за наполненностью жесткого диска загруженностью центрального процессора и использованием памяти Если возникает нехватка одного из этих ресурсов, необходимо выяснить, какие программы являются источником проблем, и устранить их. Важную роль играет контроль за обновлением программного обеспечения Старые программы часто содержат ошибки, а иногда и серьезные "бреши", влияющие на безопасность системы. Такие программы делают систему уязвимой для атак со стороны злоумышленников Одни процедуры обновления программного обеспечения являются относительно безболезненными, другие занимают очень много времени и довольно рискованны К обновлению программ следует подходить избирательно и лишь в крайних случаях прибегать к полному обновлению системы

Полный справочник по

Глава 29

Системная безопасность

В этой книге неоднократно говорилось о том, какое влияние на безопасность системы оказывают те или иные программы, процедуры и методы администрирования. Таким образом, у читателей уже сложилось определенное представление о правильных подходах к защите системы. Тем не менее эта тема столь важна, что заслуживает отдельной главы. Мы рассмотрим ее в общем контексте компьютерной безопасности, т.е. узнаем, какие виды атак существуют, как защищаться от них и как обнаруживать факты взлома системы.

Способы атак

Компьютерные взломщики знают множество способов атак, и в этом суть проблемы. Недостаточно перекрыть одну две потенциальные "лазейки", поскольку любая нетривиальная система состоит из множества компонентов, каждый из которых теоретически уязвим. Это существенно усложняет защиту компьютерных систем. Для целей нашей дискуссии разделим методы взлома на пять основных категорий: зондирование системы, дистанционные атаки, локальные атаки, непрямые атаки и атаки типа "отказ от обслуживания". У каждого из них есть особенности, позволяющие вырабо тать стратегию защиты.

Хакеры, взломщики и злоумышленники

В средствах массовой информации термин *хакер* употребляется в основном для обозначения компьютерных злоумышленников. Однако в сообществе программистов хакерами называют людей, которые занимаются программированием ради удовольствия, применяя свои знания в полезных и законных целях. Многие разработчики FreeBSD считают себя хакерами именно в этом смысле слова. Вот почему термин *хакер* не употребляется в книге. Вместо него мы говорим *взломщик* или *злоумышленник*, подразумевая человека, пытающегося взломать компьютерную систему или каким-то образом нарушить ее нормальное функционирование.

Зондирование системы

Первым этапом нападения на систему по сети обычно является ее зондирование. Как правило, взломщики применяют два типа зондирования:

- **Зондирование с целью определения типа системы.** Зная уязвимое место системы, взломщик будет зондировать большое число компьютеров, чтобы выявить среди них подходящие системы. Например, если стало известно, что в программном коде популярного сервера Apache присутствует ошибка, взломщик попытается просканировать целые сети для нахождения систем, определенные признаки которых однозначно свидетельствуют о наличии сервера Apache.
- **Детальное зондирование.** Если злоумышленник хочет взломать конкретный компьютер, он будет выполнять более глубокое зондирование, ориентированное на выявление множества потенциальных уязвимых мест. Не исключено также, что злоумышленник попробует прозондировать другие системы в той же сети в надежде взломать одну из них и воспользоваться ею в качестве плацдарма для нападения на запланированную систему.

Зондирование систем — чрезвычайно широко распространенная практика. Если компьютер постоянно подключен к Internet, его могут зондировать по нескольку раз в день. Владелец компьютера даже не догадается об этом, если не воспользуется средствами обнаружения вторжений, например не задаст правила фильтрации пакетов (см.

главу 17) Иногда факт зондирования удается установить анализируя журнальные файлы серверных программ, работающих в системе.

Зондирование чаще всего свидетельствует о враждебности намерении но не стоит сразу же раздувать скандал По одному этому факту нельзя сделать вывод о том, что имеет место попытка взлома. Доступ к потенциально уязвимому порту системы мог произойти по ошибке. Скажем, пользователь полагал что на компьютере функциони рует общедоступный сервер, или просто ввел неправильный IP адрес Не располагая явными доказательствами злого умысла (зондирование множества компьютеров с од ного IP-адреса или зондирование множества портов компьютера), вы будете выгля деть глупо, делая громкие заявления Если вас терзают нехорошие предчувствия, свя житесь с администратором сети, в которой находится подозреваемый взломщик. Что бы узнать кто отвечает за интересующий вас IP адрес воспользуйтесь командой whois

```
$ whois 172.19 20.21
```

Эта команда выдает информацию о владельце заданного IP адреса. Детали отчета зависят от сети, но в любом случае приводится контактный адрес электронной почты Если будете посылать письмо, обязательно приведите журнальные записи и другие признаки, неопровержимо свидетельствующие против соответствующего IP-адреса, а также укажите, в каком часовом поясе вы находитесь и применяется ли протокол NTP для синхронизации серверных часов с источником точного времени Используйте вежливый тон обращения, поскольку существует вероятность того, что зондирующая система тоже была взломана и человек, получающий гневное письмо, является еще большей жертвой, чем вы

Отслеживание зондирования само по себе полезно тем, что позволяет выявить наиболее популярные направления атак Например, внезапный всплеск "интереса" к UDP-порту 53 (DNS) может означать, что в программном коде DNS-сервера выявлена серьезная ошибка. Проверьте, не появились ли соответствующие анонсы на специа-лизированных Web-узлах и, если это так, обновите устаревшее программное обеспе чение

Дистанционные атаки

Согласно распространенному мнению, типичная схема атаки выглядит так: зло умышленник, находящийся за десятки или сотни километров от атакуемой системы (в другом городе, стране..), взламывает ее, чтобы украсть номера кредитных карточек, исказить вид Web-страниц и т д. Такие атаки, конечно же, случаются, но нужно смот-реть на вещи шире. Компьютер может подвергнуться нападению, даже если он не представляет для злоумышленника непосредственного интереса Как правило взлом щики используют несколько компьютеров, прокладывая "туннель" к атакуемой сис теме Это серьезно затрудняет их обнаружение. Идя по следу, пострадавшая сторона выйдет на предыдущую жертву в цепочке, но дальше цепочка обычно обрывается

Злоумышленники могут взламывать компьютеры лишь для того, чтобы запрограм мировать их на атаку других компьютеров Именно так осуществляются распределен ные атаки типа "отказ от обслуживания", рассматриваемые на с 573

Еще одна причина для взлома систем заключается в получении удовольствия от этого процесса или завоевании авторитета в своей среде Такие взломщики не ставят перед собой цель нанести вред системе, но это все равно очень часто происходит из за некомпетентности

В любом случае дистанционные атаки обычно совершаются путем использования ошибки в программном коде какого-нибудь сервера. Распространенный тип таких ошибок — это *переполнение буфера*, позволяющее злоумышленнику послать строку большего размера, чем предусмотрено сервером для хранения данных В результате

строка затирает другие данные, хранимые программой, или даже ее программный код При тщательном планировании можно заставить сервер выполнить произвольный код. Такая возможность используется взломщиками для запуска непредусмотренных серверов и получения большего контроля над системой

X Window System — это тоже сервер, а значит, он потенциально уязвим для атак Задайте правила фильтрации пакетов, ограничивающие доступ к X-серверам Первому X-сеансу обычно соответствует порт 6000, второму — 6001 и т.д Если X-сервер работает в режиме суперпользователя то риск значительно повышается, поскольку в случае взлома злоумышленник получит гораздо более широкие привилегии

Дистанционные атаки другого типа не связаны с серверами, а нацелены на ошибки клиентского программного обеспечения Например, современные Web-броузеры поддерживают языки программирования Java и JavaScript Соответственно, Web-страницы могут содержать программный код, который будет интерпретироваться Web-броузером. Это позволяет злоумышленнику включить в тело Web-страницы сценарий, выполняющий в клиентской системе несанкционированные действия. Естественно, предусмотрены средства защиты от таких сценариев, но определенный риск всегда присутствует. Одним из достоинств FreeBSD является продуманный механизм учетных записей, не позволяющий обычной программе получить полный контроль над компьютером Злоумышленник взломавший клиентскую программу, не будет иметь привилегий суперпользователя Клиентские атаки особенно распространены в среде Microsoft Windows. Примерами таких атак являются "черви" и вирусы передаваемые по электронной почте.

Повышенный риск повреждений в случае взлома объясняет почему не стоит пользоваться сетевыми клиентами работающими в режиме суперпользователя

Наконец, последний тип дистанционных атак основан на приемах *социотехники*. В этом случае злоумышленник пытается обманным путем заставить других людей (как правило, рядовых пользователей, но, бывает, на удочку попадаются и системные администраторы) сообщить ему конфиденциальную информацию или переконфигурировать свои системы нужным ему способом. Известный трюк заключается в отправке почтового сообщения, якобы от системного администратора, с просьбой сообщить пароль "для технической проверки" Пользователи должны знать, что любые запросы такого рода — жульничество. Человек, наделенный правами суперпользователя, не нуждается в паролях других пользователей для выполнения действий над их учетными записями Системным администраторам тоже не следует терять бдительность и слепо доверять инсталлируемым программам, ведь среди них может встретиться *троянский конь* — программа, которая выполняет совсем не то, о чем заявляют ее "разработчики". Чтобы не подвергать свои системы риску инсталлируйте только программы полученные из надежных источников

Локальные атаки

Многие привыкли считать, что нападения на системы осуществляются извне, но на самом деле процент локальных атак достаточно велик Этому есть два объяснения. Во-первых, в организации может завестись "диверсант", занимающийся взломом внутренних систем Понятно, что вероятность столкнуться с фактом диверсии выше в крупных системах с большим числом пользователей Во-вторых, внешний взломщик может получить доступ к локальному компьютеру через учетную запись рядового пользователя применив какой-нибудь социотехнический прием или атаковав сервер

выполняющийся в клиентской системе. В последнем случае первый этап атаки является дистанционным, зато потом локальный пользователь, сам того не ведая, становится "диверсантом".

Итак, что плохого может сделать локальный взломщик? Направлений для "диверсионной" деятельности несколько.

- **Итерационные атаки.** Взломав одну из систем, злоумышленник может воспользоваться ею в качестве плацдарма для нападения на другие компьютеры. Более того, если не предусмотреть определенные меры защиты, то о существовании такого "плацдарма" администратор узнает лишь после того, как будет атакована следующая система.
- **Незаконное пользование услугами.** Локальный взломщик может и не совершать ничего криминального, а просто осуществлять несанкционированное использование компьютера и сетевого соединения. Если говорить о собственно локальных пользователях, то такую деятельность нельзя квалифицировать как взлом. Поэтому подразумевается ситуация, когда злоумышленник действительно взломал учетную запись, но лишь для того, чтобы воспользоваться аппаратными ресурсами компьютера.
- **Кража информации.** Как и в случае дистанционных атак, локальные взломщики могут стремиться к похищению информации. В условиях жесткой конкуренции на рынке это одна из форм промышленного шпионажа.
- **Анализ сетевого трафика.** Локальный взломщик может инсталлировать *анализатор пакетов* — программу, осуществляющую низкоуровневый контроль сетевого трафика. Если в сети применяются протоколы, не поддерживающие шифрование паролей и данных, то с помощью анализатора пакетов легко организовать перехват паролей. Более того, программа сможет восстановить содержимое большинства документов, передаваемых по сети. Проблема частично решается за счет использования сетевых коммутаторов, а не концентраторов, но и они не помешают анализатору перехватывать трафик в своем сетевом сегменте.
- **Получение прав суперпользователя.** Целью локального взломщика может быть получение привилегий суперпользователя. Как правило, это нужно для достижения других целей, например для кражи информации или инсталляции анализатора пакетов.

Многие из рассматриваемых далее методик, таких как удаление ненужных программ и создание защищенных паролей, эффективны в борьбе как против локальных, так и против дистанционных взломщиков. Просто всегда нужно учитывать, какие программы имеются в распоряжении локальных пользователей. Дополнительная сложность заключается в том, что локальные пользователи часто имеют физический доступ к сети и даже к компьютеру-жертве. Если взломщика оставить одного с компьютером, он сможет украсть с него любые данные, получить права суперпользователя, извлечь жесткий диск и т.д. Вот почему важны физические меры защиты, такие как изоляция ключевых серверов в отдельных помещениях, запирание корпуса компьютера в защитном кожухе, прикручивание компьютера к полу или стене и применение паролей BIOS для предотвращения несанкционированной загрузки компьютера.

Непрямые атаки

Целью *непрямых атак* является не собственно компьютер-жертва, а связанные с ним системы. Предположим, вы организовали Web-узел, запустив на локальном компьютере сервер Apache, но при этом зависите от провайдера, предоставляющего услуги DNS внешнему миру. Тогда злоумышленник, стремящийся повредить ваш Web-узел, сможет сделать это косвенно — путем взлома DNS-сервера провайдера и переад

ресации его на Web-сервер, работающий на другом компьютере. Этот компьютер управляется (или тоже взломан) злоумышленником и содержит модифицированную версию Web-узла. Таким образом, внешним пользователям будет казаться, что Web-узел кем-то испорчен, хотя на самом деле исходный компьютер остался нетронутым.

Непрямые атаки могут применяться и для получения несанкционированного доступа к компьютеру. Допустим, на компьютере работает сервер NFS (см. главу 18), безопасность которого основывается на модели надежных узлов. В этой модели доступ к серверу предоставляется на основании IP-адреса или сетевого имени вызывающего компьютера. Если используются имена, злоумышленник может, опять-таки, взломать DNS-сервер и подделать имена, чтобы обеспечить себе "прозрачный" доступ к серверу NFS.

Всегда следует опасаться конфигураций, в которых ответственность за аутентификацию перекладывается на другие компьютеры, пусть даже они находятся под вашим непосредственным контролем. Такие конфигурации популярны в системах с централизованной аутентификацией и приводят к тому, что компьютеры, управляющие аутентификацией, становятся "лакомыми кусочками" для взломщиков.

Атаки типа "отказ от обслуживания"

Последний тип атак — это атаки, основанные на отказе от обслуживания. Такая атака нацелена на то, чтобы лишить жертву возможности использовать компьютер или сетевое соединение. В отличие от дистанционных атак, злоумышленнику нет необходимости взламывать компьютер. Ему достаточно использовать уязвимые места в программном обеспечении компьютера. Рассмотрим несколько распространенных приемов.

- **Установление слишком большого числа сетевых соединений.** Злоумышленник может установить множество соединений с сервером, не используя ни одно из них. В результате ресурсы сервера (память и центральный процессор) будут расходоваться впустую, но, главное, законные пользователи окажутся лишены доступа к серверу.
- **Манипулирование журнальными файлами.** Если злоумышленник знает о том, что определенные действия регистрируются в журнальных файлах компьютера-жертвы, он может "развернуть бурную деятельность" в надежде на то, что это вызовет непомерное разрастание журнальных файлов и соответственно перегрузку дискового канала ввода-вывода.
- **Перегрузка полосы пропускания.** Если злоумышленник располагает более широкой полосой пропускания сетевого канала, чем жертва, он может переполнить канал жертвы пакетами, чтобы помешать законным внешним пользователям получать доступ к компьютеру по сети. В такую атаку могут быть вовлечены и другие компьютеры, каждый из которых вносит свой вклад в пакетную "лавину".

От некоторых атак типа "отказ от обслуживания" невозможно защититься, по крайней мере одними лишь локальными настройками. Типичная распределенная атака, например, достигает цели независимо от конфигурации компьютера, поскольку входящий пакет в любом случае занимает часть полосы пропускания, хотите вы того или нет. Оказавшись жертвой такой атаки, свяжитесь с провайдером, чтобы он попытался блокировать поступающие пакеты у себя.

Стандартные конфигурационные настройки FreeBSD позволяют противостоять некоторым другим видам рассматриваемых атак. Например, алгоритмы ротации журнальных файлов основаны на размерах файлов, поэтому проблемы возникают только в том случае, когда злоумышленник генерирует журнальные записи слишком быстро.

Влияние таких атак минимизируется еще и благодаря изоляции журнальных файлов в разделе /var. Некоторые серверы содержат настраиваемые параметры, ограничивающие число одновременных соединений, принимаемых как от одиночного компьютера, так и в целом, что предотвращает "зависание" сервера в случае поступления большого числа запросов.

Как правило, атакам типа "отказ от обслуживания" подвергаются популярные серверы или компьютеры известных личностей, чем-то не угодивших взломщикам. Маловероятно, чтобы мишенью стала небольшая система. Скорее злоумышленник попытается взломать компьютер, чтобы превратить его в "зомби" для последующей распределенной атаки на другую систему. Меры защиты в этом случае связаны с предотвращением несанкционированного внешнего доступа.

Удаление ненужного программного обеспечения

Большинство способов атак, включая дистанционные и локальные атаки, основано на эксплуатации программных или конфигурационных ошибок в программном обеспечении, инсталлированном на атакуемом компьютере. Всегда следует учитывать риск, связанный с наличием в системе той или иной программы, и следить за появлением новых, более безопасных версий программ. Но нередко в системе присутствует больше программ, чем нужно. Риск, связанный с каждым отдельным пакетом, не велик, однако совокупный риск нельзя недооценивать. Наихудший вариант — ненужные серверы, которые, в зависимости от конфигурации сети, могут быть доступны любому пользователю Internet. В подобных случаях должны применяться брандмауэры (см. главу 17), но не менее эффективная стратегия — удаление ненужного программного обеспечения. Безопаснее всего, конечно, непосредственно удалить программу, но можно также ограничить доступ к ней или просто отключить, не выполняя деинсталляцию.

Меры по обеспечению безопасности должны быть многоуровневыми. Не следует думать, будто фильтрация пакетов и удаление ненужных серверов — взаимоисключающие понятия. Рассматривайте их как два "замка" на компьютерной "двери". Если один из них будет взломан, второй не позволит открыть дверь.

Выявление неиспользуемых программ

Выявить неиспользуемое программное обеспечение не так-то просто. На с. 552 рассматривался один из способов обнаружения таких программ. По сути это метод "грубой силы". Он предполагает проверку всех файлов в основных программных каталогах, таких как /usr/local/bin, /usr/local/sbin и /usr/X11R6/bin, и определение статуса каждой найденной программы. Понятно, что это очень утомительный процесс.

Другой подход заключается в использовании пакетной базы данных FreeBSD. Если ввести команду **pkg_info -a > packages.txt**, будет создан файл packages.txt, содержащий информацию обо всех пакетах, инсталлированных через коллекции пакетов или переносимых приложений FreeBSD. Размер такого файла бывает равен нескольким мегабайтам, что немало, зато для каждого пакета в этом файле присутствуют одна-две строки описания, позволяющие оценить его важность.

Наибольшая угроза исходит от серверов, о работе которых пользователь даже не подозревает. Описанные подходы позволяли выявить серверы, но не давали возможности определить, доступны ли они. Для поиска доступных серверов предназначена

команда netstat, рассматривавшаяся в главе 16. В частности, следующая команда формирует список всех процессов, принимающих запросы на подключение, т.е. являющихся серверами.

```
$ netstat -aA | grep -E "Proto+LISTEN"
Socket    Proto Recv Q Send-Q  Local Address      Foreign Address    (state)
c7bc3700  tcp4        0      0  *.xll             *.*                LISTEN
c7bc3b40  tcp4        0      0  *.submission      *.*                LISTEN
c7bc3d60  tcp4        0      0  *.smtp            *.*                LISTEN
c7bc21c0  tcp4        0      0  *.ssh             *.*                LISTEN
c7bc2e80  tcp4        0      0  *.telnet          *.*                LISTEN
c7bc1d80  tcp4        0      0  *.ftp             *.*                LISTEN
```

В этом примере используется шесть серверных портов: xll, submission, smtp, ssh, telnet и ftp, как видно из столбца Local Address. Эти имена определены в файле /etc/services. Если сервер контролирует порт, для которого не задано имя, вместо него будет указан номер порта. Чтобы провести дальнейшую оценку доступности сервера, воспользуйтесь командой fstat, которая выдает информацию об открытых файлах и сокетах. Отфильтруйте ее отчет с помощью команды grep, задав номер нужного сокета, указанный в столбце Socket отчета команды netstat. Проверим, к примеру, сервер submission.

```
$ fstat | grep -E "CMD|c7bc3b40"
USER     CMD          PID   FD MOUNT      INUM MODE         SZ|DW R W
root     sendma l     194    5* internet stream tcp c7bc3b40
```

Важная информация здесь приведена в первых трех столбцах, где указаны пользователь, программа, подключенная к порту, и идентификатор ее процесса. В данном случае с портом submission работает программа sendmail, которая также контролирует порт 25 (smtp). Таким образом, ничего подозрительного этот пример не выявил, если предполагать, что все перечисленные серверы действительно должны работать на компьютере.

На заметку

Порт submission в настоящее время используется мало, но в будущем ситуация может измениться. Идея заключается в том, чтобы разделить порты почтовых клиентов и серверов SMTP. Это позволило бы тщательнее контролировать взаимодействие клиентов и других серверов с почтовым сервером. В принципе, лучше закрыть порт submission от внешних пользователей с помощью брандмауэра, но разрешить локальный доступ к нему.

Для серверов, запускаемых супердемоном, команда fstat сообщает имя последнего (inetd в типичной инсталляции FreeBSD). В этом случае просмотрите файл /etc/inetd.conf, чтобы узнать, какой сервер связан с портом.

Ограничение доступа к программам

Иногда опасная программа должна работать на компьютере, но лишь санкционированные пользователи могут иметь доступ к ней. Контроль доступа к программам рядовых пользователей осуществляется иначе, чем в случае серверов. Здесь мы поговорим именно о локальном программном обеспечении, методы ограничения доступа к серверам будут рассматриваться на с. 578.

Предположим, определенная программа (или группа программ) должна быть доступна лишь некоторым пользователям. Пусть это будет Web-броузер Mozilla (это может понадобиться в серверах дистанционной регистрации, когда нужно заставить пользователей запускать броузер на локальных компьютерах для снижения нагрузки на сервер, сделав исключение для пользователей, регистрирующихся с X-терминалов).

Простейший метод ограничения доступа к программе заключается в назначении определенных прав доступа к исполняемому файлу. Рассмотрим, как это делается на примере Mozilla.

1. Создайте группу для санкционированных пользователей Mozilla (см. главу 10). Можно также использовать существующую группу, если в нее входят только пользователи Mozilla. В нашем примере это будет группа `mozusers`.
2. Добавьте в группу `mozusers` пользователей, которым разрешено запускать браузер Mozilla.
3. Поменяйте группу, которой принадлежит исполняемый файл Mozilla, на `mozusers`. Например: **`chgrp mozusers /usr/X11R6/bin/mozilla`**
4. Установите для исполняемого файла Mozilla такие права, чтобы читать или запускать его могли только владелец и члены группы `mozusers`. Это можно сделать с помощью команды **`chmod o-rx /usr/X11R6/bin/mozilla`**. Учтите, что право чтения так же важно, как и право выполнения. Если пользователь может прочитать файл, значит, он сможет создать его копию, затем поменять права доступа и запустить программу.

После выполнения описанной процедуры запускать браузер Mozilla смогут только члены группы `mozusers`. Однако не все так просто. Как было отмечено, если пользователь сумеет скопировать файл, то сможет и запустить его. Это означает, что пользователь может скопировать нужный ему исполняемый файл из другой FreeBSD-системы, обойдя предусмотренные нами ограничения. Один из способов помешать такому обходному маневру — задать параметр монтирования `noexec` для разделов, в которых пользователи имеют право записи. При наличии этого параметра FreeBSD откажется запускать файл, даже если у него установлен бит выполнения. Но тогда все каталоги, в которых пользователи имеют право записи, должны размещаться в разделах, не содержащих исполняемые файлы. Кроме того, пользователи лишатся возможности вести независимую разработку программного обеспечения, что не всегда приемлемо.

Отключение программ

Другой подход состоит в том, чтобы полностью отключить программу. Для большинства серверов процедура будет такой: 1) уничтожить серверный процесс, 2) убедиться в том, что сервер не запускается автоматически в ходе загрузки системы; и 3) убедиться в том, что сервер не указан в конфигурационном файле супердемона. В этом случае при необходимости сервер можно будет запускать вручную. Например, уходя с работы, можно запустить сервер SSH, чтобы ночью войти в систему с домашнего компьютера.

Простейший и наиболее безопасный способ отключения клиентских программ - деинсталляция, но это радикальная мера. Эффективнее просто сбросить биты выполнения у исполняемого файла. То же самое делалось и в предыдущем подразделе, но здесь подразумевается удаление *всех* битов чтения и выполнения. Таким образом, вместо команды **`chmod o-rx /usr/X11R6/bin/mozilla`** нужно ввести **`chmod a-rx /usr/X11R6/bin/mozilla`**. Чтобы вновь подключить программу, введите **`chmod a+rx /usr/X11R6/bin/mozilla`**.

Отключение программы может служить временной мерой на случай выявления потенциальной угрозы. Например, если замечено, что программа потребляет слишком много ресурсов процессора, отключите ее до тех пор, пока не будет определена причина такого поведения. Администратор может также включить в группу, которой принадлежит программа, только самого себя, что позволит ему отлаживать программу.

Удаление программ

Как уже говорилось, один из способов отключения программы — ее деинсталляция. Если программа инсталлировалась в виде пакета, воспользуйтесь командой `pkg_delete`, но для этого нужно знать точное название пакета, включая номер версии. Эту информацию можно получить с помощью команды `pkg_info` если известно имя хотя бы одного файла, входящего в пакет, например

```
# pkg_info -W /usr/X11R6/bin/mozilla
/usr/X11R6/bin/mozilla was installed by package mozilla 0 9 9 3 1
# pkg_delete mozilla-0.9.9_3,1
delete mozilla-0.9 9_3,1? y
```

Эти команды применимы и к программам инсталлированным через коллекцию переносимых приложений, хотя они работают не совсем корректно. Чтобы удалить переносимое приложение, лучше перейти в его исходный подкаталог в каталоге `/usr/ports` и ввести следующую команду

```
# make deinstall
```

Она делает по сути то же, что и `pkg delete`, но учитывает особенности инсталляции переносимого приложения. Обе команды описывались в главе 11.

Если программа инсталлируется из исходного кода в обход коллекций пакетов или переносимых приложений, то деинсталлировать ее сложнее, поскольку нет инсталляционной базы данных. В такой ситуации придется вручную отследить местонахождение всех файлов программы. Наиболее важен основной исполняемый файл программы, но многие программы поставляются со вспомогательными исполняемыми файлами, сценариями, конфигурационными файлами, документацией и т.д. Конечно конфигурационные файлы и документация не несут никакой угрозы системе, однако они занимают место на диске.

Многие инсталляционные сценарии отображают списки инсталлируемых файлов. Попробуйте повторно запустить такой сценарий чтобы получить список файлов, а затем вручную удалите их.

Возможности преодоления ограничений

Недостаток рассмотренных подходов заключается в том, что настойчивый пользователь может самостоятельно добавлять программы, копируя их с другого компьютера или даже компилируя "с нуля", а затем инсталлируя в каталог, правом записи в который он располагает. Как уже отмечалось, чтобы предотвратить такие действия пользователей, следует задать параметр `noexec` для разделов, хранящих начальные каталоги пользователей, а также для остальных разделов, открытых для записи, например `/tmp`. Причем это следует спланировать перед инсталляцией системы, иначе какой-нибудь каталог наверняка останется без внимания. Кроме того, такой подход неприменим если пользователи должны иметь возможность разрабатывать собственные программы. С другой стороны, наиболее рискованные программы принадлежат пользователю `root` и имеют установленный бит SUID. Рядовые пользователи не могут создавать файлы с такими правами доступа, если только компьютер не был взломан. Следовательно, для SUID-программ рассмотренные методики вполне подходят.

Методики отключения программ и ограничения доступа к ним стоит применять для контроля серверов и многопользовательских систем. Но они не защищают систему от зондирования, непрямых атак и многих видов атак типа "отказ от обслуживания". Можно, например, удалить все лишние программы, и все равно сетевое соединение окажется заблокированным, подвергшись распределенной атаке.

Ограничение доступа к серверам

Серверы играют слишком важную роль в компьютерном мире, поэтому они слиш ком часто подвергаются атакам Большинству локальных программ — текстовым ре дакторам, компиляторам, почтовым клиентам и т.д. не нужны специальные приви легии Как правило, с ними работают лишь локальные пользователи, поэтому исхо дящая от них угроза минимальна, даже если они содержат "бреши" с точки зрения безопасности Серверы же по своей природе доступны гораздо большему кругу поль зователей Часто они выполняются от имени пользователя root и потому их ошибки приводят к гораздо более разрушительным последствиям Таким образом серверам следует уделять особое внимание Существует множество средств защиты серверов включая пароли специальные серверные настройки программу TCP Wrappers и брандмауэры

Парольная защита серверов

Наиболее важная защита многих серверов пароли Пароли нужны не только серверам, поэтому о создании безопасных паролей мы поговорим отдельно на с. 583 Пароли используются не всеми серверами, например, Web-серверы, как правило, по зволяют всем пользователям загружать Web-страницы, и большинство почтовых SMTP-серверов принимает почту, не требуя пароля. В то же время серверы, хранящие конфиденциальные данные или предоставляющие пользователям расширенный кон троль над системой, не могут обойтись без паролей. В частности, к ним относятся почтовые серверы POP и IMAP а также большинство серверов дистанционной реги страции

Серверы, использующие пароли, полагаются на низкоуровневые механизмы аутен тификации FreeBSD, которые, в свою очередь, осуществляют обработку паролей с помощью модулей PAM (см главу 6) Это означает, что ошибка в модуле PAM может вызвать проблемы у всех серверов, выполняющих с его помощью аутентификацию пользователей. С другой стороны, сами серверы становятся менее подвержены ошиб кам аутентификации, поскольку соответствующий код вынесен в модули PAM и четко стандартизирован, к тому же регулярно проверяется разработчиками модулей

Есть серверы, обрабатывающие пароли в обход модулей PAM или использующие их лишь при определенных обстоятельствах К ним в первую очередь относится Samba (см главу 18). Сервер Samba может дополнительно принимать пароли, зашиф рованные в соответствии с требованиями внутренних протоколов SMB/CIFS сервера. К сожалению, алгоритм шифрования не совместим с тем, что применяется во FreeBSD, поэтому для аутентификации зашифрованных паролей нельзя задействовать модули PAM. Вот почему сервер Samba ведет собственный файл зашифрованных па ролей, который нужно обслуживать независимо от основного файла паролей FreeBSD Тем не менее если отключить режим шифрования, сервер будет принимать обычные пароли, вызывая для их аутентификации модули PAM.

Шифрование паролей играет особенно важную роль в случае серверов Многие протоколы, включая стандартные варианты Telnet, FTP, POP и IMAP, высылают па роли в открытом, т е незашифрованном, виде Если злоумышленник установит ана лизатор пакетов на передающем, принимающем или промежуточном компьютере, то сможет перехватывать пароли. Дальше взлом сервера будет делом техники Более того если пользователь применяет одинаковые пароли на нескольких компьютерах, зло умышленник получит доступ сразу к нескольким системам Таким образом, незашиф рованные пароли представляют собой серьезную проблему. Шифрование паролей сле дует выполнять везде где только возможно особенно если канал передачи данных не

защищен Популярным средством шифрования является протокол SSH. В нем шифруются *любые* передаваемые данные, а не только пароли Как описывалось в главе 21 этот протокол служит безопасной заменой протоколам Telnet и FTP С его помощью можно даже туннелировать данные других протоколов, изначально не выполняющих шифрование. Шифрование паролей, но не данных, поддерживается также серверами Samba и VNC

Специфические средства защиты серверов

Многие серверы содержат собственные уникальные средства защиты Рассмотрим ряд примеров

- **Ограничения IP адресов.** Серверы могут принимать или отклонять запросы на подключение от других компьютеров на основании IP-адресов. Собственно говоря, серверы типа NFS и `rlogind`, именно так и контролируют доступ, вместо того чтобы проверять имена пользователей и пароли В других серверах подобные меры защиты являются вторичными Некоторые серверы позволяют указывать имена компьютеров вместо IP-адресов, но это делает систему открытой для непрямых атак, нацеленных на взлом DNS-сервера
- **Ограничения адресов сетей** Можно ограничивать доступ на основании IP адресов сетей или, что еще более важно, на основании адресов сетевых интерфейсов Предположим, к примеру, что для упрощения процедуры администрирования маршрутизатора на нем запускается SSH-сервер Тогда, чтобы свести к минимуму риск взлома, следует сконфигурировать сервер на прием запросов только через интерфейс, подключенный к локальной сети, а не к Internet.
- **Временные ограничения.** Некоторые серверы можно конфигурировать на прием запросов лишь в определенное время суток Подобные ограничения поддерживает, к примеру, демон `xinetd` (`www.xinetd.org`), который может заменять программы `inetd` и TCP Wrappers
- **Ограничения локальной файловой системы.** Большинство файловых серверов и серверов дистанционной регистрации опирается на FreeBSD-механизмы получения доступа к файлам Если сервер запрашивает имя пользователя и пароль то пользователь получает те же привилегии, что и при регистрации с консоли (некоторые серверы, например Samba, позволяют задавать сложные правила переопределения) Другие серверы выполняются от имени определенного пользователя, не принимая во внимание регистрационные параметры конечного пользователя
- **Защита блокированием** Многие серверы отключаются от клиента или предпринимают специальные действия после определенного числа ошибок аутентификации или подозрительных команд. Например, некоторые почтовые серверы намеренно увеличивают время задержки, если отправитель пытается послать письмо по несанкционированному адресу Такая защитная мера служит эффективным заслоном против спамеров.
- **Регистрация имени дистанционного пользователя** Почти все серверы регистрируют свои действия в файле `/var/log/messages` или в других журнальных файлах Некоторые серверы можно сконфигурировать так, чтобы они фиксировали имя пользователя, при условии, что в дистанционной системе работает сервер `ident`. Эта информация не позволяет непосредственно защитить сервер, но она важна для расследования фактов взлома Если имя пользователя зафиксировано в журнальном файле, свяжитесь с администратором дистанционной системы, и ему будет проще выявить нарушителя

Некоторые из рассмотренных средств защиты можно конфигурировать вне самого сервера Например, с помощью демона cron можно модифицировать правила фильтрации пакетов, разрешая или запрещая доступ к определенным портам в зависимости от времени суток. Тем самым создаются временные ограничения даже для тех серверов, которые не поддерживают их непосредственно. Правила фильтрации позволяют реализовывать ограничения на основании IP адресов

Программа TCP Wrappers

TCP Wrappers — это стандартное средство защиты различных UNIX-систем, включая FreeBSD Программа предназначена для контроля доступа. Она вызывается супер демоном inetd перед запуском подчиненного сервера в ответ на поступивший запрос Предположим, на компьютере функционирует FTP-сервер запускаемый демоном netd Когда поступает FTP-запрос, демон inetd принимает его и тут же передает программе TCP Wrappers, которая проверяет структуру запроса, в частности исходный IP-адрес Если запрос соответствует критериям, заданным в конфигурационном файле программы, она запускает FTP-сервер и передает запрос ему. Все это занимает долю секунды, поэтому клиент ничего не замечает Если же запрос не может быть санкционирован, соединение разрывается до того, как запрос будет передан серверу.

Программа TCP Wrappers позволяет применять согласованные механизмы контроля доступа ко всем серверам, которыми управляет демон inetd (некоторые серверы вызывают программу TCP Wrappers напрямую и потому могут быть защищены независимо от демона inetd) Можно также заставить программу применять разные критерии к различным серверам Это означает, что серверы, которые планируется запускать через демон inetd, можно упростить, удалив из них код контроля доступа, поскольку эту задачу берет на себя программа TCP Wrappers.

Если демон inetd запускается с опцией W или w, он автоматически подключает программу TCP Wrappers соответственно к внутренним (предоставляются самим демоном) или внешним службам. В стандартном варианте запуска используются обе опции, поэтому программа TCP Wrappers задействуется всегда. Чтобы проверить это введите следующую команду

```
$ ps ax | grep inetd
  183  ??  Is     0:00.10 /usr/sbin/inetd  ww
```

Она выдает информацию о процессе inetd, включая командную строку в которой присутствуют опции -ww Если их нет или процесс inetd вообще отсутствует, обратитесь к главе 16, где рассказывалось о том, как запускать демон inetd

Параметры конфигурации программы TCP Wrappers хранятся в файле /etc/hosts.allow Строки, начинающиеся с символа '#' являются комментариями Остальные строки имеют такой формат

```
список_демонов : список_клиентов   параметр [ параметр    ]
```

На заметку В ранних версиях FreeBSD и в некоторых других UNIX-системах используются два управляющих файла: hosts.allow и hosts.deny Они содержат спецификации, соответственно, разрешающие и запрещающие доступ В нашем случае спецификации обоих типов задаются в одном файле

Рассмотрим назначение каждого элемента записи

- *список_демонов* В первом поле содержится список из одного или нескольких серверов Серверы могут задаваться по именам, которые определены в файле /etc/services, или с помощью различных ключевых слов, таких как ALL (соответствует всем серверам) Программа TCP Wrappers понимает также имена

дополнительных серверов, в частности `sendmail`. Имена серверов в списке разделяются запятыми или пробелами.

- *`список_клиентов`*. Во втором поле содержится список клиентов, задаваемых с использованием доменных имен, IP-адресов, шаблонов сравнения и метасимволов. Отдельные клиентские спецификации разделяются запятыми или пробелами. Кроме того, поддерживаются определенные ключевые слова, например `ALL` (соответствует любому клиенту).
- *`параметр`*. В большинстве случаев параметр равен `allow` или `deny`, что означает, соответственно, разрешение или запрет доступа. Можно указывать несколько параметров, разделенных двоеточиями, правда, в этом случае все параметры, кроме последнего, задают выполняемые команды или дополнительные проверки.

В списках демонов и клиентов поддерживается ключевое слово `EXCEPT`, позволяющее задавать исключения из правил. Например, выражение `ALL EXCEPT telnet` в списке демонов означает правило, применяемое ко всем серверам, кроме Telnet, а выражение `.luna.edu EXCEPT werner.luna.edu` в списке клиентов означает правило, применяемое ко всем компьютерам в домене `luna.edu`, кроме компьютера `werner.luna.edu`. В последнем примере используется шаблон сравнения — имя, начинающееся с точки, которое соответствует всем компьютерам домена. Существует множество различных шаблонов.

- **Отдельные доменные имена**. Компьютер можно задать по имени, например `werner luna.edu`. Такая спецификация соответствует отдельному компьютеру, адрес которого определяется путем поиска в DNS.
- **Отдельные IP-адреса**. Разрешается указывать и IP-адрес компьютера, например `172.21.102.67`.
- **Имена, начинающиеся с точки**. Доменное имя может начинаться с точки, как в случае `.luna.edu`, означая весь домен `luna.edu` со всеми поддоменами. Выражению `.luna.edu` соответствует, например, компьютер `tranquility.psych.luna.edu`.
- **Адреса, заканчивающиеся точкой**. Числовой адрес, состоящий менее чем из четырех компонентов и завершающийся точкой, интерпретируется как адрес сети класса A, B или C и обозначает все адреса этой сети. Например, выражению `172.21.` соответствуют адреса в интервале от 172.21.0.0 до 172.21.255.255.
- **Имена сетевых групп NIS**. Имя, начинающееся с символа '`@`', интерпретируется как имя сетевой группы NIS (Network Information Service — сетевая информационная служба). Чтобы эта спецификация поддерживалась, в системе должна быть инсталлирована служба NIS.
- **Сетевые маски IPv4**. Разрешается указывать IP-адрес с 4-байтовой сетевой маской, например `172.21.0.0/255.255.0.0`. Такому выражению соответствует все адреса заданной сети.
- **Сетевые маски IPv6**. Помимо обычных IP-адресов можно указывать адреса IPv6 с показателем длины префикса. Например, выражению `[3ffe:505:2:1::]/64` соответствуют все адреса от `3ffe:505:2:1::` до `3ffe:505:2:1:ffff:ffff:ffff:ffff`. Поскольку протокол IPv6 используется очень редко, необходимость в данном шаблоне почти не возникает.
- **Имена файлов**. Можно записать шаблоны сравнения в файл, указав его имя в качестве клиентской спецификации (имя файла должно начинаться с обратной косой черты). Например, в файл `/etc/badclients` можно поместить список

запрещенных клиентов. Каждая строка такого файла содержит одну клиентскую спецификацию.

Для наглядности рассмотрим несколько примеров:

```
ALL   ALL : allow
ssh : ALL : allow
ftp vnc : tranquility.psych.luna.edu,.threeroomco.com   allow
ALL EXCEPT sendmail : ALL EXCEPT 172 21  . deny
```

Первое правило встречается в начале стандартного файла `hosts.allow` во FreeBSD. Оно отключает программу TCP Wrappers, поскольку разрешает всем клиентам подключаться к любым серверам. Если необходимо, чтобы программа TCP Wrappers контролировала доступ, удалите эту строку или превратите ее в комментарий.

Второе правило разрешает всем компьютерам подключаться к серверу SSH. Такого рода правила следует использовать лишь для тех служб, которые надежно защищены и должны быть общедоступными.

Третье правило отражает более консервативную стратегию доступа. Оно применяется к серверам FTP и VNC, доступ к которым разрешен только компьютеру `tranquility.psych.luna.edu` и компьютерам домена `threeroomco.com`.

В последнем правиле демонстрируется использование ключевого слова `EXCEPT`. Оно применяется ко всем серверам, кроме `sendmail`, и запрещает доступ к ним всем компьютерам, кроме тех, у которых IP-адреса начинаются с префикса 172.21. Правила доступа к программе `sendmail` могут быть заданы отдельно.

Обычно в конце файла задается правило, определяющее стандартный порядок обработки запросов, которые не соответствуют ни одному из предыдущих правил. По умолчанию конфигурационный файл содержит следующее стандартное правило:

```
ALL : ALL \
          severity auth.info \
        : twist /bin/echo "You are  ot welcome to use %d from %h
```

В этом правиле используются расширенные параметры, но в принципе они эквивалентны параметру `deny`.[1] Таким образом, если удалить строку `ALL : ALL : allow` в начале файла, программа TCP Wrappers будет запрещать любые попытки доступа, которые не разрешены явно. Для начальной конфигурации это идеальный вариант, поскольку такая конфигурация послужит защитой от непреднамеренных ошибок в настройках серверов.

Возможности программы TCP Wrappers довольно велики. Дополнительная информация о ней приведена на man-странице `hosts_options`, где описан формат файла `/etc/hosts.allow`.

Применение брандмауэра

Брандмауэры и правила фильтрации пакетов подробно описывались в главе 17. Брандмауэры представляют собой мощное средство контроля доступа к компьютеру. Критерии контроля сходны с теми, что применяются программой TCP Wrappers. В основном проверяются IP-адреса и номера серверных портов. С помощью правил фильтрации, заданных на маршрутизаторе, можно защитить всю сеть. Кроме того, эти правила могут применяться к серверам, не взаимодействующим с программой TCP Wrappers, и даже к клиентским программам.

---

[1] Параметр `severity` включает журнальную регистрацию попыток доступа. Параметр `twist` задает команду, выполняемую вместо запрашиваемого сервера. В данном случае это команда `echo`, которая сообщает пользователю о том, что с его компьютера (переменная `%h`) нельзя получить доступ к запрашиваемой службе (переменная `%d`).

У читателей может возникнуть резонный вопрос: зачем иметь дело с программой TCP Wrappers, если можно задать не менее эффективные правила фильтрации пакетов? Таких причин три. Во-первых, правила фильтрации могут содержать скрытый изъян, который при удачном стечении обстоятельств не будет продублирован правилами программы TCP Wrappers. Тем самым программа превращается во вторую "линию обороны" системы. Во-вторых, код фильтрации пакетов, имеющийся во FreeBSD, сам может содержать ошибки, позволяющие злоумышленнику получить несанкционированный доступ в систему. И в этом случае программа TCP Wrappers также послужит вторым оборонительным рубежом. В-третьих, программа TCP Wrappers содержит расширенные параметры (не описываются в этой главе), которые невозможно продублировать правилами фильтрации пакетов.

Безопасность паролей

Пароли — чрезвычайно важные инструменты защиты, применяемые как при локальном, так и при дистанционном доступе к компьютеру. Поэтому необходимо не только создавать надежные пароли, но и обучать пользователей, чтобы пароли не стали уязвимым местом системы. Особое внимание следует уделять паролю учетной записи `root`, поскольку знание этого пароля позволяет получить полный контроль над системой.

Важность защиты паролей

Пароли можно считать "ключами" от компьютера. Вряд ли кто-нибудь захочет, чтобы ключи от его дома, автомобиля или офиса попали в чужие руки. Точно так же и пароли не должны быть известны посторонним. Чего следует опасаться при работе с паролями?

- **Простодушие.** Бывает, пользователь сам сообщает свой пароль — друзьям, сотрудникам, однокурсникам и т.д., — чтобы те смогли получить доступ к его файлам для выполнения определенных задач. При надлежащем распределении прав доступа к файлам и каталогам нет необходимости сообщать кому бы то ни было свой пароль, просто некоторые пользователи не понимают, как достичь поставленной цели более простым способом. Другой "прокол" заключается в записывании пароля на бумажке, которая клеится к монитору. Такой пароль рано или поздно станет известен кому-то еще. Последняя ошибка состоит в применении одинакового пароля на нескольких компьютерах. Взломав один из компьютеров, злоумышленник обязательно попробует ввести пароль на других компьютерах, где есть такая же учетная запись, в надежде на то, что пользователю лень запоминать несколько паролей.
- **Социотехника.** Как уже отмечалось на с. 571, злоумышленники нередко пытаются обманным путем заставить пользователей сообщить им свои пароли. На эту удочку попадаются наиболее простодушные пользователи.
- **Применение анализатора пакетов.** Еще одним средством кражи паролей являются анализаторы пакетов. Они эффективны в тех случаях, когда в сети используются протоколы, не поддерживающие шифрование паролей.
- **Кража файлов паролей.** Наибольшая удача для взломщика - украсть файл `/etc/master.passwd`, в котором хранятся пароли. Вообще-то все пароли в нем зашифрованы, но существуют довольно мощные утилиты, позволяющие взламывать наиболее очевидные пароли. Для этого зашифрованные пароли проверяются по очень большому словарю зашифрованных слов и их распро

страненных модификаций. Такие словари создаются с помощью тех же алгоритмов шифрования, которые применяются во FreeBSD при работе с пароля ми Как только утилита обнаруживает совпадение со словарным элементом пароль отгадан Чтобы противостоять словарным атакам необходимо выбирать пароли, которые наверняка не встретятся ни в одном словаре

- **Угадывание.** Злоумышленник может попытаться просто угадать пароль. Такое случается не только в кино — слишком часто в качестве паролей пользователи выбирают собственные регистрационные имена.
- **Подглядывание.** В многолюдных рабочих помещениях, например в универси тетских компьютерных лабораториях, взломщик может сидеть неподалеку и подглядывать за тем, как пользователь вводит пароль. Тут главное не терять бдительность, поскольку риск есть всегда На этот случай рекомендуется выби рать пароли, вводимые как можно более быстрым движением пальцев
- **Троянские кони, маскирующиеся под программы регистрации** В общедоступных компьютерных помещениях злоумышленник может запустить ложную про грамму регистрации, которая в точности имитирует работу стандартной сис темной утилиты регистрации; при этом пользователю кажется, будто он непра вильно ввел пароль (последний не отображается на экране). В действительно сти же программа фиксирует введенный пароль, после чего завершается, передавая управление системной утилите регистрации Ничего не подозревающий пользователь считает, что случайно ошибся, хотя на самом деле у него только что украли пароль.

Против некоторых из этих атак наиболее эффективными являются администра тивные меры Например, можно удалить или отключить сервер Telnet, заменив его сервером SSH, что сделает применение анализатора пакетов малоэффективным (чтобы полностью исключить риск, нужно переконфигурировать или отключить ряд других серверов, таких как POP или XDMCP). Другим атакам противостоять намного сложнее Даже самый идеальный пароль можно узнать проследив за движением пальцев по клавиатуре

В правильно сконфигурированной системе трудно украсть файл паролей, но такие случаи были Злоумышленник, которому удалось заполучить файл паролей крупной многопользовательской системы, с помощью программы типа Crack сможет в считан ные часы взломать наиболее слабые пароли Шифрование паролей на уровне сетевых протоколов тоже не панацея, поскольку иногда можно выделить такие пароли и применить к ним те же средства взлома, что и в случае файла паролей Невозможность полностью избавиться от риска означает, что пользователи должны особенно тща тельно подходить к выбору паролей и периодически менять их

Создание паролей

Труднее всего взламывать пароли, представляющие собой случайные наборы букв цифр и знаков препинания К сожалению, запоминать такие пароли весьма затрудни тельно, ведь нет никаких "зацепок", ассоциаций Администраторы, заставляющие пользователей запоминать такие пароли, часто замечают, что это дает *противоположный* эффект, т.е. безопасность ослабевает Почему? Потому, что пользователи записывают пароли на бумаге, так сказать, "на всякий случай" А раз пароль где-то записан, то его проще украсть. Таким образом, идея случайных паролей прекрасная в теории не всегда оправдывает себя на практике.

В качестве компромисса можно предложить двухэтапную процедуру составления пароля. Сначала нужно придумать *основу*, которая сама по себе легко запоминаема и не может служить надежным паролем Далее необходимо модифицировать эту основу

так, чтобы полученный результат нельзя было встретить в словаре взломщика Распространенными вариантами основ являются комбинации из двух слов ("good" + "boy" = goodboy), а также акронимы, получаемые из легко запоминаемых фраз, например yiwttd (от "yesterday I went to the dentist" — "вчера я был у стоматолога") Маловероятно, чтобы такие комбинации присутствовали в словаре, хотя можно вести поиск основ первого типа, последовательно группируя короткие слова

На заметку В показанных примерах получены относительно короткие основы, поскольку в некоторых системах длина пароля ограничивается восемью символами. Впрочем, во FreeBSD допускаются более длинные пароли, а чем длиннее пароль, тем лучше Считается, что 8 символов — *минимальный* безопасный размер пароля.

Внимание Не используйте показанные здесь основы, так как после публикации книги они могут быть занесены в словари взломщиков. Всегда составляйте собственную основу.

Полученная основа может подвергаться различным модификациям, направленным на увеличение ее размера и уменьшение вероятности взлома путем словарной атаки Рассмотрим несколько возможных модификаций

- **Добавление символов**. Добавьте к основе числа или знаки препинания, желательно в случайной позиции Например, из goodboy может получиться g#od7boy, а из yiwttd — yi0wtt<d.
- **Изменение регистра символов** Поменяйте регистр символов случайным образом, например g#Ood7bOY или YI0wtT<d Некоторые системы не различают регистр символов в паролях, поэтому такая методика не всегда применима. Стандартные пароли FreeBSD, вводимые посредством модулей PAM, чувствительны к регистру, а зашифрованные пароли SMB/CIFS — нет
- **Изменение порядка следования букв на противоположный** Если есть основа, полученная путем объединения двух слов, имеет смысл инвертировать одно из них. Например, g#Ood7bOY превратится в g#Ood7YOb

Идея модификации состоит в том, чтобы сделать основу максимально непохожей на слова, которые могут встретиться в словаре взломщика Что если взломщик располагает словарем, содержащим комбинации распространенных коротких слов? Тогда пароли типа goodboy будут разгадываться достаточно легко. Но вряд ли в таком словаре учтены еще и все возможные вариации регистра, порядка букв и т д.

Замечания по поводу учетной записи root

Учетная запись root играет чрезвычайно важную роль в системе и ее пароль — самое ценное, что может "добыть" взломщик Поэтому к безопасности такого пароля должны предъявляться повышенные требования Никогда не пересылайте пароль суперпользователя через незашифрованное соединение. В частности, никогда не следует выполнять команду su (запрашивает привилегии суперпользователя) в сеансе Telnet или при работе в среде X Window System в дистанционном режиме (если только данные сеанса не пересылаются через SSH-туннель). Необходимо также регулярно менять пароль, чтобы в случае его взлома у злоумышленника осталось не так много времени для разрушительных действий

Естественно, системным администраторам свойственны те же слабости что и рядовым пользователям К примеру, администратору точно так же трудно запоминать произвольные пароли. Но, учитывая его задачу — защищать систему, следует смириться с определенными неудобствами

Обнаружение взлома

Описывавшиеся выше меры предосторожности, безусловно, важны, но стопро центно безопасных систем не бывает Что бы ни предпринимал администратор и как бы опытен он ни был, всегда существует вероятность того, что кто-нибудь прорвется сквозь все защитные барьеры Это может произойти из-за наличия неизвестной (для администратора, но не для взломщика) ошибки в программном коде сервера или не правильной конфигурации сервера либо иной программы. Другие потенциальные причины – кража пароля или несанкционированный физический доступ к компью теру Поскольку нельзя полностью исключить вероятность взлома, следует предусмот реть план его обнаружения. Чтобы быстро выявить непрошеного гостя, необходимо прежде всего быть бдительным, регулярно просматривать журнальные файлы и при менять средства обнаружения вторжений, в частности программу Tripwire. Важно также знать, какие действия предпринимать если факт взлома системы подтвердился

Общие признаки

К счастью, далеко не все взломщики достаточно опытны для того, чтобы умело скрыть свое присутствие. Часто применяемые утилиты взлома учетной записи `root` оставляют после себя следы, включая модифицированные утилиты, работающие не так, как оригиналы, модифицированные конфигурационные файлы, искажающие нормальное функционирование системы и т д Иногда вторжение в систему можно выявить на основании определенных признаков, на появление которых всегда следует обращать внимание

- **Сбои программ**. Если программа внезапно перестает работать, выясните, в чем дело К примеру, если стандартная программа, такая как `ls` или `cat`, вдруг да ет сбой, не исключено что взломщик модифицировал ее или заменил троян ским конем.
- **Ненормальное поведение программ**. Признаком взлома может служить не только сбой программы Для UNIX вообще характерно, что программы типа `ifconfig`, `grep` и `netstat` по-разному работают на разных платформах Даже во FreeBSD программы время от времени подвергаются модификации. Напри мер, опции, поддерживаемые во FreeBSD 5.0, могут отличаться от опций во FreeBSD 4.6. Злоумышленник может заменить одну из программ модифициро ванной версией, заимствованной из другой UNIX-системы или из другой вер сии FreeBSD. Тогда пользователю будет казаться, будто результаты работы программы или ее опции внезапно изменились
- **Изменения в структуре распределения ресурсов** Иногда злоумышленники взла мывают системы для того, чтобы воспользоваться ресурсами процессора, сете вым каналом и т д В главе 28 рассказывалось о том, как проверять использо вание системных ресурсов Если обнаруживается внезапное изменение в пове дении системы выясните не является ли это следствием деятельности взломщика
- **Несоответствие паролей**. Если пользователь приходит к вам и заявляет, что он похоже, забыл свой пароль, выясните, так ли это Спросите у пользователя когда он последний раз входил в систему, а затем сопоставьте полученную ин формацию с содержимым файла `/var/log/auth.log`. Возможно, пользователь правильно помнит свой пароль, просто злоумышленник сменил его.

- **Подозрительные действия пользователей** Следите за попытками регистрации в системе в непривычное время. Если известно, что пользователь в отпуске и не имеет доступа к сети, но кто-то зарегистрировался под его именем с локального терминала, скорее всего учетная запись пользователя взломана.
- **Пустые строки в журнальных файлах** Взломщики иногда удаляют строки из журнальных файлов и даже сами журнальные файлы в попытке замести следы. Если обнаруживаются подобные факты, выясните причину. Например, поищите с помощью команды `find` файлы, созданные или модифицированные именно в тот период, которому соответствуют отсутствующие записи. Уделите особое внимание системным исполняемым и конфигурационным файлам.

Если замечен один из указанных признаков или появилось иное отклонение от нормальной работы FreeBSD, то это *может* свидетельствовать о взломе системы. Подчеркнем слово "может". Сами по себе все эти признаки ни о чем не говорят. Та или иная программа действительно могла быть обновлена, пользователь мог на вполне законных основаниях запросить большой объем ресурсов, он на самом деле мог забыть пароль или, уехав в отпуск, оставить компьютер работать. Суть в том, чтобы определить истинную причину подозрительных признаков. Если обнаружен посторонний пользователь, вошедший в систему под чужим именем с общедоступного терминала, или странный сервер, подключенный к порту, который никогда ранее не использовался, — только в таких случаях следует предпринимать соответствующие меры, о которых рассказывается на с. 591.

Программа Tripwire

FreeBSD — сложная операционная система, и выполнять ее мониторинг вручную очень трудно. Это чрезвычайно утомительный и трудоемкий процесс. Собственно, с целью автоматизации подобной работы и были изобретены компьютеры. Неудивительно, что существуют программы, позволяющие контролировать безопасность системы. Одна из наиболее популярных среди них — Tripwire (`www.tripwire.org`). Она доступна в разделе `security` коллекции переносимых приложений FreeBSD. Существует также целый ряд других утилит, описываемых на с. 590, которые не столь мощны, как Tripwire, но полезны в определенных ситуациях.

Принципы функционирования

Работа программы Tripwire основана на вычислении контрольных сумм важных системных файлов. После инсталляции программы ее нужно периодически запускать, чтобы она сравнивала записанные контрольные суммы с фактическими. Если выявляется несовпадение значений или отсутствие важного файла, программа сообщает о проблеме. Таким образом, если злоумышленник взломал компьютер и модифицировал какие-то системные файлы, при очередном запуске программа Tripwire обнаружит это. Как правило, программа вызывается раз в день демоном `cron` и высылает отчет по электронной почте ответственному лицу.

Если в системе делаются важные изменения, например модифицируется какая-то программа или конфигурационный файл, необходимо обновить базу данных программы Tripwire, чтобы она не подняла ложную тревогу.

Опытный взломщик, знающий о присутствии программы, может предпринять контрмеры, например откорректировать ее базу данных, удалив сведения об изменениях. Однако в последних версиях программы ее базы данных и конфигурационные файлы шифруются для предотвращения подобных модификаций. Впрочем, лучше всего скопировать файл базы данных на съемный носитель (дискеты недостаточно, но Zip-диск подойдет), чтобы затем время от времени сравнивать резервную копию с ис-

ходным файлом, убеждаясь в отсутствии отличий Учтите, что сравнение необходимо делать перед обновлением базы данных после чего следует заменить резервную ко пию новым вариантом.

Tripwire — полезное средстао обнаружения вторжений особенно если инсталлировать программу сразу после установки и конфигурирования самой операционной сис темы Рекомендуется установить FreeBSD, сконфигурировать систему без подключе ния к Internet, инсталлировать и сконфигурировать программу Tripwire и только затем подключить компьютер к Internet. Это сведет к минимуму риск взлома, который мо жет произойти до инициализации базы данных Tripwire. К сожалению, программу проще всего инсталлировать в виде переносимого приложения, а для этого необходи мо подключиться к Internet Так что с практической точки зрения имеет смысл ин-сталлировать программу сразу после подключения к Internet. Если установить про грамму на компьютер, который уже взломан, она просто запишет контрольные суммы взломанных файлов и не заметит ничего подозрительного

Инсталляция

Если ввести **`make install`** в исходном каталоге Tripwire, сценарий инсталляции спросит, куда записывать исполняемые файлы потребует ввести две идентификаци онные фразы (общесистемную и локальную) для шифрования баз данных и сгенери рует исходную базу данных. На выполнение некоторых операций может уйти не сколько минут, так что запаситесь терпением Программе придется прочесть большое число системных файлов и сгенерировать для них контрольные суммы

На заметку Идентификационная фраза — это пароль, состоящий из нескольких слов. Программа Tripwire использует две такие фразы. Одна из них применяется для шифрования файла правил, а другая — для шифрова ния базы данных содержащей контрольные суммы

В процессе инсталляции база данных программы записывается в каталог `/var/db/tripwire` Ее название соответствует доменному имени компьютера, на пример `werner.luna.edu.twd` Этот файл хранится в зашифрованном и сжатом виде поэтому не пытайтесь просмотреть его в текстовом редакторе. Конфигурационные файлы программы Tripwire находятся в каталоге `/usr/local/etc/tripwire`.

- **`twcfg.txt`**. В этом файле задаются общие установки программы, такие как расположение файлов баз данных и параметры отправки почтовых отчетов
- **`tw.cfg`**. Программа не работает с файлом `twcfg.txt` напрямую. Вместо этого она читает зашифрованную версию файла, называющуюся `tw.cfg` Этот файл шифруется с помощью общесистемной идентификационной фразы
- **`twpol.txt`**. Для важных системных файлов программа Tripwire вычисляет не сколько типов контрольных сумм и сравнивает их со значениями, хранящими ся в базе данных. В *файле правил* задаются проверяемые файлы и типы кон трольных сумм, вычисляемых для каждого из них.
- **`tw.pol`**. Это зашифрованная версия файла правил с которой собственно и работает программа Tripwire

Изменение конфигурации

Вероятнее всего, изменения придется вносить в файл `twpol.txt`. Когда выполня ется начальное конфигурирование программы Tripwire (в ходе выполнения команды **`make install`**), на экране могут появляться сообщения об ошибках, касающиеся не существующих файлов и каталогов. Эти сообщения будут повторяться при каждом за

пуске программы, если не превратить в комментарии строки файла twpol.txt, ссылающиеся на соответствующие файлы и каталоги. Признаком комментария служит символ '#', который добавляется в начало строки

Если удалить ссылку на файл или каталог из файла правил программы Tripwire, а затем создать этот файл или каталог, программа не будет его контролировать, пока пользователь не введет ссылку снова.

Формат файла правил программы довольно сложен. Полностью он описан в документе Tripwire Policy Guide (/usr/local/share/doc/tripwire/policyguide.txt). Файл состоит из нескольких частей. Вначале задаются переменные среды. Например, следующая строка сообщает программе, где искать файл базы данных.

```
TWDB="/var/db/tripwire"
```

В остальных строках определяются классы файлов. Программа выполняет разные проверки в зависимости от того, к какому классу принадлежит файл. Следующая строка, к примеру, задает имя класса исполняемых файлов

```
SEC_BIN      = $  ReadOnly) ;      # исполняемые файлы, которые
                                   # не должны меняться
```

В дальнейших определениях исполняемые файлы можно помечать как $SEC_BIN. Эти определения сообщают программе, какие файлы и каталоги следует проверять. Таковы, например, правила обработки исполняемых файлов самой программы

```
# Исполняемые файлы Tripwire.

   ulename = "Tripwire Binaries
  severity = $ SIG_HI


  $(TWBIN,/siggen                          > $ SEC_BIN  ,
  $(TWBIN /tripwire                        > $(SEC_BIN) ;
  $(TWBIN),twadmin                        -> $(SEC_BIN) ;
  $ TWBIN)/twprint                        -> $(SEC_BIN  ,
}
```

Определение состоит из двух блоков. Первый блок заключен в круглые скобки. В нем задается название правила и уровень его важности. Второй блок заключен в фигурные скобки, и в нем используется оператор присваивания ->, сообщающий программе, как обрабатывать указанные файлы и каталоги. В примере это не показано, но если строке файла предшествует восклицательный знак, правило является *исключающим*, т.е. программа не станет проверять объект.

Инициализация и обновление базы данных

Программа Tripwire автоматически инициализирует свою базу данных в процессе инсталляции. Если в конфигурацию вносятся существенные изменения, имеет смысл полностью переинициализировать базу данных. Это делается так.

```
# /usr/local/sbin/tripwire --init
```

Показанная команда заставляет программу повторить процесс инициализации, в ходе которого вычисляются контрольные суммы всех файлов, заданных в файле правил, и создается новая база данных с контрольными суммами. Для аутентификации следует ввести общесистемную и локальную идентификационные фразы.

Программа поддерживает две операции обновления. Можно обновлять отдельно базу данных, а можно — базу данных вместе со всей конфигурацией. Первый вариант необходим, когда модифицируется или инсталлируется пакет FreeBSD либо конфигурационный файл. Соответствующая команда такова.

```
# /usr/local/sbin/tripwire   update
```

Если же вносятся изменения в конфигурацию самой программы, нужна другая команда. Она принимает в качестве аргумента текстовый файл правил `twpol.txt` и преобразует его в зашифрованную форму, а также обновляет базу данных программы Вот эта команда:

```
# /usr/local/sbin/tripwire  -update policy \
/usr/local/etc/tripwire/twpol.txt
```

В этом случае тоже нужно указать общесистемную и локальную идентификационные фразы. Если команда обнаружит изменения в каких-либо файлах кроме конфигурационных, она откажется обновлять базу данных Чтобы изменить эту установку, задайте опцию `--secure-mode low`

На выполнение каждой из трех команд может уйти несколько минут

Применение

После того как создана точная база данных программы Tripwire, запускайте команду `tripwire` с опцией `--check` для сравнения текущего состояния системы с тем состоянием, которое зафиксировано в базе данных

```
# /usr/local/sbin/tripwire --check
```

Распространенный подход заключается в том, чтобы создать задание демона `cron` (см главу 6) для выполнения проверок на регулярной основе, скажем, в 3 часа ночи, когда загруженность системы минимальна Программа Tripwire запишет отчет в поток `stdout`, и демон `cron` направит этот отчет по электронной почте пользователю вызвавшему программу, т.е в данном случае пользователю `root`

Другие программы обнаружения взлома

Tripwire — одно из наиболее мощных средств FreeBSD, предназначенных для обнаружения вторжений. Но существуют и другие аналогичные программы, которые мы рассмотрим лишь вкратце

- **AIDE (Advanced Intrusion Detection Environment)** — решает задачи, сходные с Tripwire. Она доступна не только в виде переносимого приложения, но и пакета.
- **`chkrootkit`** — выявляет наличие на компьютере известных утилит, предназначенных для взлома учетной записи `root` Запустите эту программу, если программа Tripwire еще не инсталлирована и есть подозрение что компьютер взломан
- **`clog`** — регистрирует информацию, которая обычно не фиксируется для сетевых соединений. Сама по себе эта программа не является средством обнаружения вторжений, но ее можно использовать вместе с утилитами мониторинга журнальных файлов, такими как `logcheck` или SWATCH для выявления подозрительной деятельности в системе.
- **Doctor Web** Эта программа инсталлируется из пакета `drweb` и представляет собой детектор вирусов во FreeBSD Она наиболее полезна для выявления вирусов Windows, заразивших файлы на сервере Samba Вспомогательная программа `drweb-sendmail` проверяет на наличие вирусов входящую почту
- **`fwlogwatch`** Выявление подозрительных действий путем анализа журнальных файлов — слишком утомительное занятие Эта программа формирует сводку журнальных записей, сгенерированных утилитой фильтрации пакетов (см главу 17).
- **`logcheck`** Подобно `fwlogwatch` эта программа осуществляет мониторинг журнальных файлов, но ведет учет не только действий фильтра пакетов

- **Snort** — усложненный анализатор пакетов. Программы такого рода могут использоваться взломщиками, но они же полезны и как средство административной диагностики сети. Программа Snort предназначена для выявления подозрительных действий в сети. Она способна обнаруживать взлом не только того компьютера, на котором выполняется. Применение сетевых коммутаторов снижает эффективность программы, поэтому имеет смысл инсталлировать ее на брандмауэрах.
- **SWATCH** — программа мониторинга журнальных файлов. Она может высылать по почте отчеты о важных событиях или создавать метафайл, содержащий только подозрительные журнальные записи.
- **YAFIC**. Название этой программы расшифровывается как "Yet Another File Integrity Checker" — "еще одно средство проверки целостности файлов".

Все эти программы инсталлируются из раздела `security` коллекций пакетов или переносимых приложений. Просмотрите этот раздел, чтобы узнать, какие еще программы доступны в системе. Существуют десятки утилит, предназначенных для выявления "брешей" в программах, проверки качества паролей, добавления новых методов аутентификации и т.д.

Внимание! Некоторые средства защиты с успехом могут использоваться и взломщиками. Особенно это касается анализаторов пакетов типа Snort, программ взлома паролей и сканеров портов. Помните, что, применяя такую программу, вы рискуете сами нарваться на обвинения в противоправной деятельности и оказаться под следствием. Прежде всего необходимо строго соблюдать правила конфиденциальности, принятые в данной организации, и в особых случаях запрашивать письменное разрешение у администратора сети.

Что делать при выявлении взлома

Взлома системы можно избежать, но если все усилия оказались тщетными и вы столкнулись с фактом взлома, возникает сакраментальный вопрос: что делать? Вкратце процедура такова:

1. **Отключитесь от Internet**. Если система взломана, она может служить плацдармом для нападения на другие компьютеры. Следовательно, необходимо немедленно отключить ее от Internet.
2. **Создайте резервную копию**. Очевидно, имеет смысл создать резервную копию системы. В крайнем случае заархивируйте пользовательские данные. Резервная копия может послужить уликой в суде, а ее изучение позволит собрать информацию о том, как произошел взлом.
3. **Исследуйте причину взлома**. Попытайтесь выяснить, как злоумышленник проник в систему. К сожалению, это проще сказать, чем сделать, поскольку многие взломщики почти не оставляют после себя следов. На специализированных Web-узлах, посвященных безопасности, рассказывается о том, какие фрагменты журнальных файлов могут служить признаками взлома. Если точную причину установить не удалось, узнайте, какие "заплаты" появились с момента последнего обновления системы и какие именно "бреши" они ликвидируют.
4. **Очистите компьютер**. Удалите FreeBSD с компьютера. Если схема разбивки на разделы позволяет, это можно сделать на следующем этапе путем переформатирования инсталляционных разделов, оставив пользовательские данные нетронутыми. Если же каталог `/home` не хранится в отдельном разделе, придется созда-

вать новые разделы. К сожалению недостаточно удалить лишь те файлы которые были изменены злоумышленником, поскольку он мог искусно замаскировать свою работу Проще предположить что изменению подверглись все системные файлы и удалить их

5 **Переинсталлируйте систему** Переинсталлируйте FreeBSD с компакт-диска. Это позволит гарантировать чистоту системы, при условии полного переформатирования разделов FreeBSD (исключение может составлять каталог `/home`).

6. **Выполните обновления и установите "заплаты".** Чтобы предотвратить возникновение аналогичных проблем в будущем, устраните все "дыры", выявленные в п 3 Необходимо также укрепить общую безопасность системы, особенно если не удалось выявить конкретную "дыру", через которую "пролез" взломщик. Например если программа Tripwire еще не инсталлирована, сделайте это сейчас, и переключитесь с протокола Telnet на SSH. Наиболее безопасный способ выполнения обновлений — загрузить необходимые пакеты на другой компьютер а затем перенести их в восстанавливаемую систему на дискетах или других съемных носителях

7 **Проверьте пользовательские файлы** Не исключено, что взломщик оставил после себя троянского коня, например SUID-программу, владельцем которой является пользователь `root` Воспользуйтесь командой `find` для поиска исполняемых файлов в каталоге `/home`, например **`find /home -perm +111`**

8 **Поменяйте пользовательские пароли** Если каталог `/etc` был удален, то в этом шаге нет необходимости, но если вы пытались сохранить учетные записи, скопировав базы данных пользователей, не забудьте поменять пароли. Возможно, злоумышленник проник в систему, украв пароль. В таком случае восстановление старых паролей снова сделает систему уязвимой.

9 **Восстановите функционирование системы** Только после того, как будут выполнены все изменения и улучшения, можно подключаться к Internet. Конечно, некоторые программы проще инсталлировать, уже имея подключение, поэтому постарайтесь по возможности скопировать файлы с другого компьютера. Если без подключения не обойтись, временно задайте максимально жесткие правила фильтрации пакетов и загрузите обновления сразу после восстановления соединения

Если антивирусная программа обнаружила вирус, например Windows вирус в каталоге Samba, не нужно выполнять перечисленные выше действия, переинсталлируя систему Достаточно уничтожить вирус либо удалить и заново инсталлировать зараженные файлы

Выполнение приведенной процедуры позволит свести к минимуму риск если не прошеный гость решит нанести повторный визит в процессе восстановления системы или вскоре после этого. В течение некоторого времени соблюдайте повышенные меры безопасности, поскольку злоумышленник может, не подозревая что его раскрыли снова попытаться проникнуть в систему. В таком случае появится шанс выследить его. К сожалению, компьютерные преступления все еще широко распространены, несмотря на то что взлом компьютерных систем преследуется по закону в большинстве стран Просто у правоохранительных органов не хватает ресурсов для расследования каждого случая. Зачастую рассчитывать на помощь можно лишь в случае значительного материального ущерба

Ни в коем случае не пытайтесь платить злоумышленнику той же монетой, отвечая атакой на атаку. Ни к чему хорошему это не приведет. Internet — не место для вендетты Вы лишь привлечете к себе внимание правоохранительных органов, для которых все нарушители одинаковы

Резюме

Системная безопасность — чрезвычайно важный аспект администрирования FreeBSD Существуют различные способы нападения на системы: от атак типа "отказ от обслуживания" до непосредственного взлома системы У каждого способа есть еще несколько разновидностей что существенно усложняет задачу администратора Безопасность системы зиждется на простых правилах: удалять ненужные и регулярно обновлять используемые программы, особенно серверы Важными превентивными мерами являются также корректное конфигурирование программ и создание защищенных паролей. К сожалению, даже самым опытным администраторам приходится сталкиваться с фактами взлома систем Важно знать какие признаки свидетельствуют об имевшем место взломе и как на него реагировать. Чем быстрее администратор примет спасительные меры, тем меньший ущерб успеет нанести злоумышленник

Полный справочник по

Глава 30

Компиляция программ

В главе 11, где описывался процесс инсталляции программного обеспечения во FreeBSD, рассматривались два вида программ: *пакеты* и *переносимые приложения*. Пакеты — это заранее скомпилированные исполняемые файлы, которые легко инсталлировать. Для переносимого приложения необходим сценарий, который загрузит исходный код, применит к нему "заплаты", специфичные для FreeBSD, скомпилирует программу и инсталлирует ее, зарегистрировав в пакетной базе данных. Поэтому переносимые приложения инсталлируются дольше, зато можно получить более качественный результат в плане производительности или надежности, поскольку программа компилируется с учетом особенностей конкретной системы.

Но не всегда есть возможность инсталлировать программу в формате пакета или переносимого приложения. В таком случае приходится компилировать и инсталлировать программу вручную. В этой главе мы рассмотрим процесс компиляции, узнаем, какие средства для этого применяются и как инсталлировать полученные программы.

Принципы компиляции

Программы, инсталлируемые во FreeBSD, в основном доступны в виде пакетов или переносимых приложений, поэтому необходимость в самостоятельной компиляции программ возникает редко. К этому приходится прибегать в тех случаях, когда программа нестандартна или имеет более новую версию, чем та, что доступна в системном архиве. Если в исходный код программы вносятся изменения, ее тоже нужно перекомпилировать.

Процесс компиляции

Большинство программ пишется на *языках высокого уровня*, таких как C, C++, Pascal, Modula-2 или FORTRAN. Ядро FreeBSD написано преимущественно на C, а системное программное обеспечение — на C и C++. Но все эти языки непонятны центральному процессору, который работает с *машинными*, или *двоичными*, кодами (иногда между этими терминами проводят различие в зависимости от способа представления кода). Двоичный, или бинарный, код представляет собой последовательность чисел, описывающих выполняемые процессором операции. Двоичные коды одного семейства процессоров (скажем, Intel x86 и совместимые модели компаний AMD, VIA и Transmeta) несовместимы с двоичными кодами другого семейства (допустим, PowerPC компаний Motorola, IBP и Apple). Даже в пределах семейства возможны вариации. Например, процессоры Pentium поддерживают команды, отсутствующие в более ранних моделях 80486. В связи с этим программы, написанные на языках высокого уровня, должны преобразовываться в низкоуровневый машинный код для конкретного процессора или семейства процессоров. Такой процесс называется *компиляцией* программы.

Процесс компиляции состоит из двух или трех этапов:

1. Каждый программный файл транслируется из исходного языка на язык ассемблера, т. е. в особую форму машинного кода. Вместо чисел, представляющих процессорные операции, в ассемблере используются мнемонические коды.
2. Каждый ассемблерный файл преобразуется в *объектный файл*, содержащий машинный код. Сам по себе объектный файл еще не является программой.
3. Все объектные файлы объединяются в единое целое, и к ним подключаются *библиотеки*, содержащие коды наиболее распространенных функций. В результате такого процесса, называемого *компоновкой*, получается исполняемая программа.

Некоторые компиляторы преобразуют исходный код непосредственно в объектный файл, объединяя первый и второй этапы На каждом этапе возможны определенные модификации Например, библиотеки могут компоноваться *статически*, т.е. включаться в итоговый программный файл, или *динамически*, т.е. сохраняться в отдельных файлах Во втором случае размер исполняемого файла получается меньше и соответственно снижается загруженность памяти, поскольку программы совместно работают с одними и теми же библиотеками. Но вместе с тем необходимо, чтобы каждая такая библиотека присутствовала в системе, в которой запускается программа Обновление библиотек может потребовать перекомпиляции зависящих от них программ. Одним из преимуществ коллекций пакетов и переносимых приложений FreeBSD является то что такие зависимости контролируются автоматически, но все равно обноиение отдельной программы няи библиотеки может вызвать цепочку проблем.

Большинство программ состоит из нескольких исходных файлов. Такой подход упрощает совместную разработку программ и сокращает усилия по перекомпиляции программы в ходе разработки· если в программу вносятся изменения, то перекомпиляции подвергаются только модифицированные файлы. Для слежения за тем, какие файлы отредактированы, и с целью автоматизации процесса компиляции применяются специальные утилиты Одна из наиболее популярных называется `make`. Именно она, в частности, используется для компиляции и инсталляции переносимых приложений, как описывалось в главе 11 Многие пакеты исходных кодов содержат файл `Makefile`, управляющий процессом компиляции. В более сложных программах имеется сценарий `configure`, создающий нужный файл `Makefile` после изучения параметров системы

За и против

Мотивы для компиляции программы бывают разными

- **Доступность исходного кода**. Некоторые программы не предлагаются в виде пакетов или переносимых приложений FreeBSD, но доступны их исходные коды В этом случае можно попробовать скомпилировать программу во FreeBSD
- **Оптимизация с учетом особенностей системы**. Приступая к компиляции, программист обычно редактирует файл `Makefile` либо передает сценарию `configure` или утилите `make` параметры, позволяющие оптимизировать процесс компняяции для конкретного процессора
- **Внесение модификаций** Имея исходный код, можно вносить в него изменеиия Некоторые из них, например редактирование сообщений в строке состояния или надписей в меню, даже не требуют глубоких знаний в области программирования.
- **Доверие**. Когда есть исходныи код программы, можно просмотреть его и убедиться в том, что программа не является троянским конем Не стоит, конечно, подозревать асе программы, имеющиеся во FreeBSD, но если безопасность системы играет особенно важную роль, имеет смысл проверять наиболее ответственные программы. Компилируя такие программы самостоятельно, вы будете знать, что они не преподнесут никаких "сюрпризов"

Разумеется, в некоторых случаях без глубоких знаний в области программирования не обойтись. Процедуры оптимизацни тоже в немалой степенн зависят от конкретного программного пакета, поэтому приходится тщательно штудировать документацию

К недостаткам самостоятельной компиляции системных программ можно отиести следующее

- **Затраты времени** Даже компняяция программы через коллекцию переносимых приложений отиимает немало времени Затраты временн еще больше возрас-

тут, если придется самостоятельно обрабатывать все файлы, да к тому же устанавливать "заплаты", обеспечивающие компиляцию или инсталляцию программы во FreeBSD. Подумайте, стоят ли такие затраты того небольшого выигрыша в производительности, которого можно добиться за счет оптимизации программы.

- **Отсутствие информации в пакетной базе данных.** Информация о самостоятельно скомпилированной и инсталлированной программе не заносится в пакетную базу данных. Это затрудняет удаление или обновление программы. Переносимые приложения FreeBSD лишены такого недостатка.

Средства компиляции

Чтобы иметь возможность компилировать программы, нужно инсталлировать в системе определенные инструментальные средства. Все они доступны в коллекциях пакетов и переносимых приложений FreeBSD.

Компиляторы машинных языков

Основным инструментальным средством является, конечно же, сам *компилятор* — программа, преобразующая код на языке высокого уровня в ассемблерный или машинный код. У каждого языка программирования свой компилятор, поэтому для компиляции программ на языках, скажем, Pascal и C нужны разные средства. Во FreeBSD используется коллекция GCC (GNU Compiler Collection), включающая компиляторы важнейших языков, в том числе C и C++. Эта коллекция инсталлируется по умолчанию. Другие компиляторы инсталлируются отдельно в виде соответствующих пакетов.

Для компиляции программ может понадобиться *ассемблер* — программа, преобразующая мнемокод в объектный код. Ассемблер является зависимым модулем компилятора, поэтому программа `sysinstall` инсталлирует их в связке.

Для объединения объектных файлов в единый исполняемый файл нужен *компоновщик*. Во FreeBSD используется компоновщик `ld`. Он является частью коллекции GCC и потому присутствует в большинстве систем.

В документации к программе, как правило, приводятся сведения о том, как ее компилировать. В крайнем случае посмотрите расширения файлов. Файлы с расширениями `.c` и `.h` написаны на языке C, с расширениями `.cxx`, `.c++` и `.cpp` — на C++, а с расширениями `.p` и `.pas` — на Pascal. Проверьте также файл `Makefile`, где указано, какая программа вызывается для компиляции: `gcc` (C), `cpp` (C++) или другая.

Вспомогательные библиотеки и файлы заголовков

Большинство программ зависит от одной или нескольких библиотек функций. При компиляции программы все необходимые библиотеки должны присутствовать в системе. Сценарий `configure` проверяет наличие библиотек и в случае отсутствия хотя бы одной из них выдает сообщение об ошибке. Такую проверку делает и сам компилятор на этапе компоновки программы. Иногда об отсутствии библиотеки становится известно только в момент запуска программы. Это происходит, когда программа компилируется в одной системе, а запускается — в другой. Сообщения об отсутствии библиотек бывают весьма загадочны, так что внимательно прочитайте документацию к программе, чтобы узнать, какие библиотеки ей необходимы.

С библиотеками тесно связаны файлы заголовков. Они содержат информацию, определяющую взаимодействие программ с библиотеками. Чтобы скомпилировать программу, использующую библиотеку функций, нужно иметь соответствующий файл заголовков. При отсутствии такого файла выдается сообщение об ошибке компиляции. Если повезет, из текста сообщения можно будет определить, какой именно файл необходим. Дополнительные подсказки следует искать в документации. В большинстве случаев файлы заголовков инсталлируются вместе с библиотеками, но иногда библиотеки содержатся в отдельных пакетах разработки.

В некоторых случаях файлы заголовков присутствуют в системе, но компилятор не может найти их, поскольку сконфигурирован для просмотра других каталогов. Здесь есть два выхода: либо переконфигурировать компилятор, либо создать символическую ссылку на файл в том каталоге, где этого ожидает компилятор.

Универсальные системы разработки

Как уже отмечалось, компиляцией программ обычно управляет утилита `make`. Она проверяет наличие объектных файлов и, если они отсутствуют или устарели, создает новые файлы. Затем утилита компонует все эти файлы, получая исполняемый файл. Детали процесса задаются в файле `Makefile`. Утилита `make` инсталлируется в системе по умолчанию, поэтому о ней можно не беспокоиться. У нее, впрочем, есть разновидности, например `gmake`. Если программа должна компилироваться с помощью альтернативного варианта утилиты `make`, поищите его в коллекции пакетов или коллекции переносимых приложений.

Для редактирования исходного кода программы необходим текстовый редактор. Подойдет самый простой редактор, такой как Emacs, `vi` или NEdit. Некоторые редакторы имеют специальные режимы, упрощающие редактирование текстов программ. В частности, в режиме *синтаксических выделений* определенные фрагменты программного файла отображаются другим шрифтом или цветом (например, комментарии — курсивом, а ключевые слова — зеленым цветом).

Отладчик — это утилита, организующая запуск программы под контролем разработчика. С помощью отладчика можно наблюдать за тем, как меняется содержимое программных переменных. Отладчики чрезвычайно полезны для выявления и устранения ошибок. Во FreeBSD чаще всего применяется отладчик `gdb`. Существуют графические надстройки к нему, такие как `ddd` и `kdbg`, упрощающие отладку, особенно для тех, кто не знаком с командой `gdb`.

CVS (Concurrent Versioning System — система параллельного управления версиями) позволяет организовать совместную работу над программными проектами. С ее помощью программисты могут отмечать начало и завершение работы над файлом. Рядовые пользователи применяют CVS для получения последних версий исходных кодов программ. Ее же используют для управления исходными кодами ядра и коллекцией переносимых приложений, как описывалось в главах 11 и 12.

Интегрированная среда разработки — это оболочка, объединяющая такие инструменты разработки, как компилятор, отладчик и текстовый редактор. Во FreeBSD доступны оболочки Anjuta (рис. 30.1), gIDE, KDevelop, Moonshine, MOTOR, Qt Designer, QtEZ и `xwpe`. Большинство из них являются графическими и поддерживают конкретные наборы элементов управления. Некоторые оболочки предназначены лишь для определенных языков программирования. Существуют также программы, которые могут функционировать в качестве интегрированной среды разработки, в частности Emacs. Для компиляции и инсталляции существующих программ среда разработки не нужна. Необходимость в ней возникает при написании собственных программ или внесении радикальных изменений в исходный код.

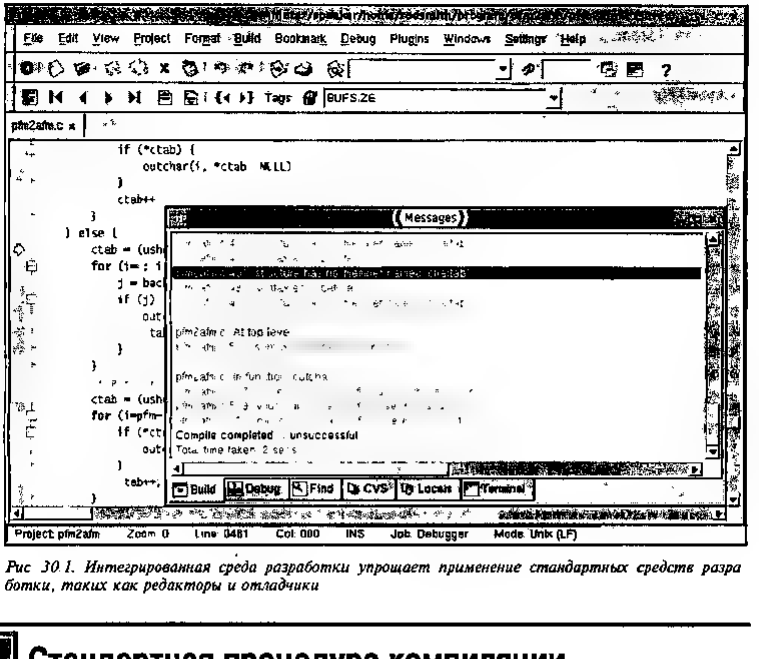


Рис 30 1. Интегрированная среда разработки упрощает применение стандартных средств разработки, таких как редакторы и отладчики

Стандартная процедура компиляции

Детали компиляции программ существенно различаются хотя определенные этапы остаются общими для большинства пакетов. Мы выделим четыре этапа: получение программы, чтение документации, конфигурирование программы и собственно компиляция. Отдельно следует вести разговор об инсталляции программ, поскольку перед инсталляцией пользователь имеет возможность решить, стоит ли продолжать дальше В процессе инсталляции необходимо следить за тем, куда помещаются файлы и не перезаписывают ли они существующие файлы других программ.

На заметку В этом разделе ничего не говорится о внесении изменений в исходный код программы. Если требуется модифицировать программу, воспользуйтесь текстовым редактором или интегрированной средой разработки

Получение программы

На первом этапе необходимо получить программу. Как правило, ее можно найти на официальном Web-узле разработчика или в FTP-архиве. В качестве образца мы возьмем программу MLVWM 0 9 1 (Macintosh Like Virtual Window Manager), доступную по адресу `http://www2u biglobe.ne.jp/~y-miyata/mlvwm html` Она достаточно велика, чтобы требовать конфигурирования, и в то же время довольно быстро

компилируется, даже на старых компьютерах. Программа представляет собой диспетчер окон, смоделированный на основе старых систем Macintosh.

Программа MLVWM доступна как в виде пакета, так и в виде переносимого приложения, поэтому инсталлировать ее по описываемой ниже схеме нет необходимости

Перейдите на Web-страницу, посвященную программе, и загрузите ее исходный код, щелкнув на соответствующей ссылке. В результате будет получен файл `mlvwm091.tar.gz`. Чтобы просмотреть имена файлов, содержащихся в архиве введите **`tar tvfz mlvwm091.tar gz | less`**. Архивы исходных кодов обычно хранят все файлы в подкаталоге имя которого соответствует названию программы в данном случае `mlvwm091`

Перейдите в каталог, где будет храниться исходный код программы Многие администраторы используют для этих целей каталог `/usr/src`, но можно выбрать, например, каталог `/usr/local/src` или свой начальный каталог Команды компиляции доступны рядовому пользователю при условии, что он имеет право записи в выбранный каталог, тогда как для выполнения инсталляции нужно быть суперпользователем. Теперь введите **`tar xvfz`** *`/путь/`***`mlvwm091.tar.gz`**, где *`/путь/`* — это каталог, в котором хранится архив Будет создан подкаталог `mlvwm091`

Всегда проверяйте содержимое `tar`-архива, используя опцию `--list` (или `-t`), прежде чем извлекать из него файлы (опция `extract`, или `x`). Если разработчик создал архив без вложенного подкаталога, файлы будут извлечены в текущий каталог

Чтение документации

После извлечения файлов перейдите в подкаталог исходного кода и введите `ls`, чтобы получить список файлов Поищите среди них файлы документации; их имена обычно записываются прописными буквами или имеют расширение `.txt` В случае программы MLVWM таких файлов три. `CHANGELOG` `CONFIGURATION` и `README` (есть также одноименные файлы с расширением `.jp` это их японские эквиваленты). Файл `CHANGELOG` включается во многие пакеты и содержит описание изменений по сравнению с предыдущими версиями. Просмотрите его, если обновляется существующий пакет. В файле `CONFIGURATION` описан формат пользовательского конфигурационного файла программы `~/.mlvwmrc`. На этапе компиляции его можно проигнорировать. А вот фаил `README` содержит информацию о том как компилировать и инсталлировать программу Для некоторых программ процедура инсталляции описана в отдельном файле `INSTALL` Аналогичная информация обычно приводится и на Web-узле программы

Документацию следует читать даже несмотря на то что в программах обычно используются стандартные процедуры компиляции, инсталляции и конфигурирования Все-таки нюансы могут отличаться, и в документации даются необходимые пояснения по этому поводу. Нередко приводятся сведения о том, как оптимизировать работу программы за счет тех же параметров компиляции, подключить или отключить тот или иной модуль, задать расположение файлов данных и т д Программа MLVWM в этом смысле довольно проста у нее нет специальных опций

Конфигурирование программы

При компиляции любого более-менее крупного программного проекта необходимо выполнить определенную команду, чтобы создать файл `Makefile` Во многих пакетах для этого предусмотрен сценарий `configure` Для запуска сценария следует ввести **`./configure`** Сценарии поддерживает дополнительные опции, позволяющие указать

конкретную модель процессора, включить поддержку определенных функциональных возможностей и т.д. В программе MLVWM применяется другой подход: файл `Makefile` создается утилитой `xmkmf`. Во FreeBSD эта утилита входит в пакет `imake`. В документации к MLVWM рекомендуется ввести **`xmkmf -a`**, чтобы сконфигурировать программу. На выполнение команды уйдет несколько секунд. Сценарий `configure` обычно выполняется немного дольше, так как он проверяет большее число системных параметров.

Если в документации к программе есть описание конфигурационных опций, посмотрите, не подойдут ли они для данного случая. Соответствующую информацию можно также поискать в Internet. У программы MLVWM таких опций нет. Файл `Makefile` разрешается редактировать и напрямую, но для этого требуется хорошо знать установки используемого компилятора.

Компиляция программы

Как правило, компиляция программы осуществляется с помощью команды **`make`**. Утилита `make` проверяет файл `Makefile`, используя содержащиеся в нем команды для вызова компилятора, ассемблера и компоновщика. На создание исполняемого файла может уйти от нескольких секунд до нескольких часов, в зависимости от размера проекта и скорости компьютера. Для каждого компилируемого файла утилита `make` выдает на экран описание выполняемой команды. Описание обычно начинается с `cc`, `gcc` или названия другого компилятора и содержит большое число опций. Нередко число опций столь велико, что описание занимает несколько строк экрана.

Иногда появляются предупреждения компилятора наподобие следующего:

```
borders.c:729: warning: `win' might be uninitialized in this function
```

Эти предупреждения довольно безобидны. Они информируют программиста о ситуации, которая, хотя и не помешает компиляции или запуску программы, но может привести к нежелательным последствиям. Программист, как правило, лучше компилятора знает, что в действительности делает программа, поэтому на предупреждения можно не обращать внимания. С другой стороны, сообщения об ошибках вызывают прекращение компиляции, например:

```
borders.c: In function `SetUpFrame':
borders.c:254: `lcp' undeclared (first use in this function)
```

Затем выдается несколько предупреждений, после чего появляется следующий текст:

```
*** Error code 1

Stop in /home/rodsmith/Projects/mlvwm091/mlvwm.
*** Error code 1
```

Все эти сообщения указывают на то, что возникла проблема, сделавшая продолжение компиляции невозможным. Если для редактирования исходного текста программы не хватает знаний, придется отказаться от попытки инсталлировать программу. В противном случае постарайтесь выяснить, в чем причина ошибки. Здесь сообщается о том, что в строке 254 файла `borders.c` содержится ссылка на необъявленную переменную `lcp` (я намеренно поменял имя переменной `lp` на `lcp`, чтобы вызвать ошибку компиляции). Это может послужить исходным пунктом для отладки программы, хотя вопросы отладки выходят за рамки книги.

Иногда сообщения об ошибках генерируются при компиляции вполне функциональной программы. Наиболее вероятная их причина — отсутствие необходимой библиотеки, поэтому в первую очередь узнайте, какие библиотеки требуются для работы программы. Не исключено также, что в программе используются определенные механизмы, не поддерживаемые имеющимся компилятором или текущей версией FreeBSD. Такие системные зависимости трудно обойти, и это одна из причин, по ко

торой существует коллекция переносимых приложений FreeBSD Каждый элемент коллекции включает "заплаты", модифицирующие программу таким образом чтобы она правильно компилировалась во FreeBSD

Инсталляция скомпилированных программ

Если программа скомпилирована успешно т.е не было выдано никаких сообщении об ошибках можно приступать к ее инсталляции В документации к пакету обычно ука зывается только команда инсталляции, но желательно выполнить ряд предварительных проверок, чтобы снизить вероятность повреждения существующих программ. После ин сталляции программы нужно убедиться в том, что она корректно работает

Выявление конфликтующих пакетов

Как описывалось в главе 11, во FreeBSD ведется база данных с информацией об инсталлированных файлах и пакетах Наличие такой базы данных позволяет выпол нять автоматизированное обслуживание пакетов, например с помощью команды `pkg update`, которая в процессе обновления пакета удаляет старые файлы Такая проверка не проводится, если пакет инсталлируется из исходного кода в обход кол лекции переносимых приложений FreeBSD. Хуже того, такой пакет может перезапи сать уже имеющиеся программные файлы, инсталлированные другим способом. Когда это происходит, пакетная база данных начинает содержать неточную информацию Хорошо, если перезаписываемая программа является старой версией новой програм мы и их файловая структура почти полностью совпадает. Но ведь программы могут не иметь никакого отношения друг к другу за исключением случайно совпадающих имен некоторых файлов

Теоретически можно просмотреть файл `Makefile` и определить, какие файлы и куда именно инсталлируются Однако на практике получить исчерпывающую инфор мацию трудно Нужно очень хорошо разбираться в структуре файла `Makefile`, кото рый содержит многоуровневые ссылки на переменные и даже обращения к файлам `Makefile` в подкаталогах

Более эффективный вариант — поискать в инсталляционном каталоге исполняе мые файлы и выяснить, нет ли в системе файлов с аналогичными именами В случае программы MLVWM результат будет таким

```
$ find ./  perm +0111 ! -type d
./man/mlvwm.man
 /man/mlvwm. man
  mlvwm/mlvwm
```

В списке присутствует только что скомпилированный главный программный файл `mlvwm/mlvwm`, а также man-страницы программы, для которых, оказывается, установлен бит выполнения Теперь можно поискать, есть ли еще где-то в системе файл с именем `mlvwm` В случае отрицательного результата можно с уверенностью сказать, что инстал ляция программы не приведет к перезаписи существующих исполняемых файлов

От предыдущих инсталляций часто остаются также файлы документации. Поищите в исходном каталоге программы подкаталоги `man`, `doc`, `documentation` и `т п`, а затем сверьте их с каталогами `/usr/local/man` и `/usr/X11R6/man` чтобы выяснить, нет ли дубликатов

Если программа ни разу до этого не инсталлировалась в системе, дубликатов ско рее всего не будет Если же присутствует предыдущая версия программы, инсталлиро вавшаяся из исходных кодов, то вполне допустимо записать новую версию поверх нее Следует лишь учитывать два нюанса

- **Дублирующиеся файлы** Иногда разработчики меняют каталоги инсталляции файлов В таком случае в системе могут появиться две копии исполняемого файла. одна, скажем, в каталоге `/usr/X11R6/bin`, другая — в каталоге `/usr/local/bin`. Старая программа будет доступна одновременно с новой, что вызовет путаницу Решение заключается в том, чтобы вручную удалить старые программные файлы и все их вспомогательные файлы
- **Устаревшие файлы** Если в новой версии программы два файла объединены в один, при инсталляции новой версии появляются неиспользуемые файлы. Они не причиняют никакого вреда, просто занимают место на диске. Такие файлы тоже нужно удалять вручную

Всех этих проблем можно избежать, используя коллекции пакетов и переносимых приложений FreeBSD В долгосрочной перспективе они позволяют существенно упростить процедуры инсталляции Хотя, если речь идет об одной-двух программах возникающие трудности вполне преодолимы.

Если нет уверенности в том, стоит ли инсталлировать программу, запустите ее для тестирования в каталоге компиляции В крайнем случае просто удалите компиляционный каталог

Процесс инсталляции

В файле `Makefile` определен специальный целевой модуль `install`, описывающий процедуру инсталляции Она соответствует процедуре, которая принята для коллекции переносимых приложений FreeBSD, только в случае программ, инсталлируемых после компиляции исходных кодов, пакетная база не задействуется Типичный процесс инсталляции выглядит так:

```
# make install
installing in ./mlvwm.
/usr/bin/install -c -s mlvwm  usr X11R6/bin m vwm
install in mlvwm done  .
installing in ./sample_rc.
install in sample rc done
installing in  /man.
install in man done...
```

Поскольку в ходе инсталляции обычно происходит запись файлов в системные каталоги, такие как `/usr/local/bin` или (как в предыдущем примере) `/usr/X11R6/bin`, команду `make install` необходимо вводить в режиме суперпользователя Большинство программ выводит сообщения о происходящих событиях. Это может быть информация о каталогах, из которых берутся файлы, или перечень всех инсталлируемых файлов. Такие сведения очень полезны при отладке

Зафиксируйте на будущее, какие файлы инсталлировались Если впоследствии будет принято решение удалить программу, воспользуйтесь имеющимся списком для поиска и удаления файлов программы Удобнее всего создавать такой список с помощью утилиты `script`. Просто введите **`script record.txt`**, чтобы сохранить в файле `record.txt` все команды, выполняемые в интерпретаторе По окончании инсталляции введите **`exit`**, чтобы завершить работу утилиты

Команда `make install` доступна в большинстве архивов исходных кодов, но все же не во всех В некоторых небольших пакетах требуется вручную копировать соответствующие файлы в каталоги программ, например `/usr/local/bin`. Иногда предусматриваются инсталляционные сценарии типа `install` Подробности следует искать в документации к программе

Проверка полученного результата

Самой надежной проверкой инсталлированного пакета является его запуск. В случае программы MLVWM необходимо поменять вызов имеющегося диспетчера окон на `mlvwm` в файле `.xinitrc`, `.xsession` или другом сценарии запуска X-среды после чего перезагрузить X-сервер. Если все пройдет успешно, значит, инсталляция удалась Если нет, придется выполнять отладку Программа, которая скомпилирована корректно, должна и запускаться без проблем Возникающие трудности могут быть связаны с неявными ошибками или несовместимостью с FreeBSD, что потребует немалых знаний в области отладки.

Если говорить о запуске диспетчера MLVWM во FreeBSD 5.0, то для нормальной работы программы необходимо создать файл `.mlvwmrc` в начальном каталоге пользователя, иначе программа будет функционировать в минимальном варианте. Скопируйте файл `sample_rc/Mlvwmrc` под именем `~/.mlvwmrc`, а затем отредактируйте его так, как описано в файле `CONFIGURATION` В свою очередь файл `.mlvwmrc` ссылается на несколько графических файлов, многие из которых хранятся в подкаталоге `pixmap` исходного каталога MLVWM. Скопируйте их в соответствующий системный каталог, например `/usr/local/include/X11/pixmaps` (возможно, его придется создать) Такого рода нюансы зависят от конкретного пакета, поэтому обращайте внимание на сообщения об ошибках или странное поведение программы при первом запуске. Если что-то пойдет не так, обратитесь к документации или выполните поиск в Internet, чтобы узнать, какие еще действия нужно предпринять для обеспечения оптимальной работы программы.

В некоторых случаях нужно проверить конфигурационные настройки, например то, кому принадлежит программный файл и каковы права доступа к нему Иногда программа должна принадлежать пользователю `root` и иметь установленный бит SUID Соответственно, следует ограничить круг пользователей, которым разрешен запуск такой программы. Это делается за счет механизма группового доступа, как описывалось в главе 29

Резюме

Коллекции пакетов и переносимых приложений FreeBSD содержат огромное множество программ в легко инсталлируемом виде. В то же время эти коллекции имеют определенные ограничения Некоторые программы недоступны ни в виде пакетов, ни в виде переносимых приложений, или же их необходимо сконфигурировать либо модифицировать так, как не позволяют системные механизмы инсталляции В таких случаях приходится компилировать программу из исходного кода предоставляемого разработчиком. При этом нужно потратить время на изучение инструкций по компиляции и инсталляции каждого пакета. Инсталлируемые подобным образом пакеты не доступны для системных утилит управления пакетами, поскольку они не регистрируются в пакетной базе данных Конечно, любой системный администратор FreeBSD должен знать, как инсталлировать пакет исходных кодов, однако проще работать с готовыми пакетами или переносимыми приложениями

Полный справочник по

Глава 31

Написание сценариев

В книге неоднократно упоминались *сценарии интерпретатора команд*. Например в главе 6 описывался особенно важный класс таких сценариев, управляющих запуском системы. Сценарии — это разновидность программ, написанных на *интерпретируемом языке*. У каждого из пользовательских интерпретаторов команд свой язык. От администратора FreeBSD не требуется быть экспертом в области написания сценариев, однако владение соответствующими навыками будет очень полезно по двум причинам. Это позволит, во-первых, редактировать существующие сценарии, во-вторых – самостоятельно писать сценарии автоматизации утомительных процедур, которые в противном случае пришлось бы регулярно выполнять вручную.

В этой главе мы сначала рассмотрим, что собой представляют сценарии и в чем их отличие от компилируемых программ. Затем будет рассказано о распространенных языках сценариев, после чего можно будет поговорить о важнейших элементах сценариев, таких как вызовы внешних команд, переменные, условные выражения и циклы. Приведенная информация позволит читателям понимать (и в случае необходимости — модифицировать) существующие сценарии, а также создавать собственные.

Компилируемые и интерпретируемые программы

Сценарии отличаются от программных файлов тем, что их можно открыть в текстовом редакторе и просмотреть, чтобы определить назначение, при условии знания языка сценариев, разумеется. В случае исполняемого файла сделать нечто подобное не представляется возможным, поскольку после компиляции такой файл содержит машинный код.

Машинный и текстовый код

Центральные процессоры компьютеров понимают язык чисел. Чтобы заставить процессор выполнить определенную операцию, например сложить два числа, нужно передать ему эти два числа, а также еще одно число, являющееся кодом операции. В процессе передачи чисел процессору и записи полученного результата в память могут задействоваться дополнительные числовые операторы. Для человека все это выглядит как беспорядочный набор чисел. Для процессора же такой язык программирования называется *машинным кодом*.

Человеку сложно читать машинные коды, еще труднее создавать их. Поэтому были разработаны языки программирования более высокого уровня. Их можно классифицировать по степени отдаленности от команд типичного процессора. Наиболее низкий уровень имеет язык ассемблера. В нем поддерживается однозначное соответствие между кодами языка (*мнемониками*) и машинными инструкциями. Ассемблер всегда адаптирован к определенному типу процессора, поэтому ассемблерную программу, написанную, допустим, для PowerPC, нельзя выполнить на компьютере x86. На ступеньку выше в классификации находится язык C, который содержит много низкоуровневых конструкций и в то же время настолько далек от машинного языка, что его программы приходится *компилировать* для конкретного процессора, т.е. достаточно сложным способом преобразовывать в машинные коды, как описывалось в главе 30. Остальные языки, в том числе Pascal, FORTRAN и C++, а также языки сценариев, имеют еще более высокий уровень.

Современные компьютерные языки жестко специфицированы и контекстно независимы, в отличие от естественных языков. В конце концов, языки программиро

вания являются средством управления компьютером, а не средством общения И хотя они содержат определенное число слов английского языка, зарезервированных для уз кого применения, до естественного языка им очень далеко. Чтобы эффективно вла деть языком программирования, нужно изучить его лексикон, синтаксис и грамматику

Интерпретаторы

Как преобразовать программу, написанную на языке высокого уровня типа C или Pascal, в низкоуровневый машинный код? Один из подходов рассматривался в гла ве 30 Он заключается в компиляции программы Большинство крупных программ FreeBSD, включая системное ядро написано именно на языках высокого уровня и было скомпилировано. Однако это не единственный способ получения машинного кода Второй подход — *интерпретация* программы, когда процессор строка за строкой анализирует значение каждого ключевого слова и каждой переменной высокоуровне вой программы непосредственно в ходе ее выполнения Такой подход требует приме нения *интерпретатора*, а не компилятора Интерпретатор — это программа (сама по себе компилируемая), которая читает исходный файл и немедленно выполняет эквивалентные машинные инструкции По сути интерпретатор динамически преобразует исходный код в машинный Это различие иллюстрируется на рис. 31.1.

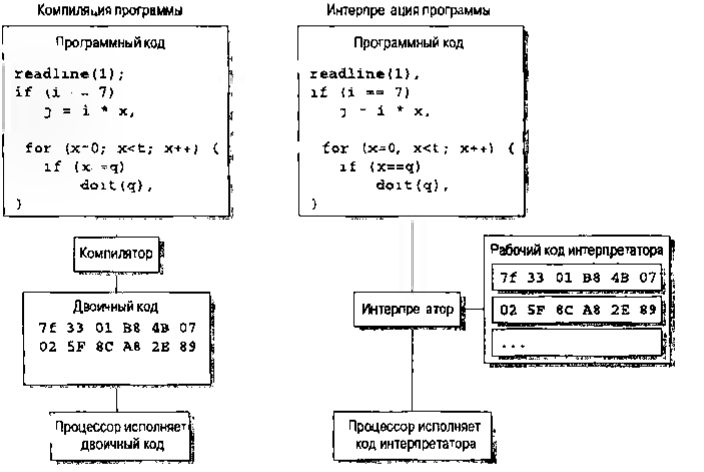


Рис 31 1 Компилятор создает программу, которая непосредственно выполняется процессором, тогда как интерпретатор динамически создает исполняемый код, передаваемый процессору

Языки сценариев обычно являются интерпретируемыми, хотя это не синонимы Некоторые высокоуровневые языки допускают как компиляцию, так и интерпрета цию и при этом не считаются языками сценариев Во FreeBSD и других UNIX-системах сценарии чаще всего начинаются такой строкой

```
#!/bin/sh
```

Первые два символа этой строки служат кодом по которому ядро узнает что это — сценарий В большинстве языков сценариев символ # обозначает комментарий, вследствие чего интерпретатор игнорирует строку. Оставшаяся часть строки определяет интерпретатор, в котором выполняется сценарий В данном случае это /bin/sh

Некоторые языки занимают промежуточное место между компилируемыми и интерпретируемыми языками. Их программный код преобразуется в формат, который нельзя считать полностью скомпилированным, т.е. это — не машинный код. Зато его проще интерпретировать чем исходный код, поэтому он будет быстрее выполняться

За и против

Различие между компилируемыми и интерпретируемыми языками связано с тем какое количество времени процессора уходит на обработку программы В первом случае компилятор тратит много времени, создавая исполняемый файл, который затем обрабатывается процессором. Если применяются эффективные методики программирования и компилятор в достаточной степени оптимизирован, можно получить программу которая будет быстрее выполняться процессором В случае интерпретируемого языка время на компиляцию вообще не расходуется, зато интерпретатор вынужден тратить время на обработку каждой строки программного файла что не позволяет оптимизировать программу Такая программа будет выполняться медленнее, чем скомпилированный аналог

Интерпретаторы удобно применять при разработке программ поскольку можно проверять изменения, не перекомпилируя программу, что существенно экономит время. В то же время их явный недостаток — скорость выполнения программ Таким образом, интерпретаторы подойдут для написания программ "по быстрому" и программ, которые предполагается часто модифицировать. Первое объясняет, почему языки сценариев столь популярны среди системных администраторов, второе — почему на этих языках пишутся сценарии запуска системы

Распространенные языки сценариев

В настоящее время существуют десятки языков программирования у каждого из которых есть свои преимущества. Например на языке ассемблера опытный программист может писать чрезвычайно эффективный код, скажем, для приложений реального времени. Популярный язык C служит промежуточным звеном между языками ассемблера и компилируемыми языками более высокого уровня, такими как C++ или Pascal У языка Perl мощные средства обработки строк. Даже среди языков сценариев существует большое разнообразие Рассмотрим наиболее популярные языки сценариев.

- **Языки интерпретаторов команд** Интерпретаторы, позволяющие пользователям вводить команды в терминальном окне или в текстовом сеансе регистрации поддерживают собственные языки сценариев. Можно создать файл, содержащий набор команд, и заставить интерпретатор выполнить этот файл. Кроме того, доступны распространенные программные конструкции, описываемые в следующих разделах. Именно на таких языках пишутся сценарии запуска системы
- **Perl.** Язык сценариев Perl (Practical Extraction and Report Language — практический язык создания отчетов и выборок) широко применяется во FreeBSD Он многое позаимствовал из других языков и славится своими командами обработки строк.

- **Python**. Названный в честь популярной комедийной группы "Monty Python" этот язык содержит удобные средства обработки строк и часто используется как "мостик", соединяющий две несвязанные программы
- **Tcl**. Язык Tcl (Tool Command Language — язык инструментальных команд) предназначен для создания сценариев-оболочек, взаимодействующих с другими программами. Библиотеки команд Tcl могут также вызываться непосредственно из программ, что превращает Tcl в универсальный язык сценариев. Расширение Tcl — Tk — позволяет Tcl-сценариям иметь графический интерфейс.
- **awk**. Названный по первым буквам фамилий своих авторов — Альфреда Ахо (Alfred Aho), Питера Вайнбергера (Peter Weinberger) и Брайана Кернигана (Brian Kernigan), — этот язык сценариев часто используется для анализа строк а также в сочетании с потоковым редактором sed, что позволяет писать программы, модифицирующие текстовые файлы. Многое из awk было позаимствовано в Perl, но Perl — более мощный язык, поэтому он применяется шире

У каждого языка есть преданные сторонники, уверенные в том, что именно этот язык — наилучший. Мы не будем здесь дискутировать по этому поводу, а просто выберем один из языков для использования в последующих примерах. В данном случае это будет язык интерпретатора команд, доступный во всех FreeBSD-системах.

Как уже отмечалось, у каждого интерпретатора команд свой язык сценариев. В этих языках реализованы особенности самих интерпретаторов, поэтому они немного отличаются друг от друга. Не всегда сценарий tcsh будет работать, скажем, в интерпретаторе Korn. Но желательно писать сценарии, не зависящие от интерпретатора, чтобы их можно было легко адаптировать к различным конфигурациям системы. Путь к исполняемому файлу интерпретатора задается в первой строке сценария. В приводимых ниже примерах используется интерпретатор /bin/sh.

Во FreeBSD под именем /bin/sh доступен исходный интерпретатор Bourne, но в других UNIX-системах возможны нюансы. Например, в Linux /bin/sh — это символическая ссылка на интерпретатор bash (Bourne-again shell), который является расширенным вариантом интерпретатора Bourne. Таким образом, после копирования в другую UNIX систему сценарий может начать работать иначе. Если сценарий ориентирован на запуск в среде Bourne, то проблем не должно возникнуть. А вот Linux-сценарии, зависящие от специфических особенностей интерпретатора bash, который "маскируется" под именем sh, во FreeBSD могут работать некорректно, если не поменять начальную ссылку (вместо /bin/sh — /usr/local/bin/bash).

Запуск и останов сценария

Помимо двух начальных байтов, идентифицирующих текстовый файл как сценарий, есть еще одни признак: у сценариев интерпретатора команд устанавливается бит выполнения. Это означает, что для запуска сценария достаточно ввести его имя в командной строке. Если бит выполнения отсутствует, имя сценария придется указывать в качестве аргумента при вызове соответствующего интерпретатора. Таким образом, создав новый сценарий, установите для него бит выполнения такой командой:

```
$ chmod a+x myscript.sh
```

Многие сценарии завершаются достаточно бесцеремонно": они просто останав ливаются. Однако наиболее правильный способ завершить сценарий — выполнить команду `exit`:

```
exit 0
```

Эта команда сообщает интерпретатору о том, что выполнение сценария завершено Даже если далее идут другие команды, они игнорируются. Самое главное, команда `exit` принимает аргумент, являющийся *кодом завершения* сценария. С помощью этого кода можно сообщать об успешном завершении сценария другим программам или сценариям, вызывающим его. Нормально завершающаяся программа должна возвра шать код 0. Значения больше нуля свидетельствуют об ошибке. При желании можно использовать коды завершения для указания на различные виды ошибок. В после дующих разделах будет показано как проверять такие коды завершения в условных конструкциях сценариев.

Использование внешних команд

Чаще всего сценарии применяются для вызова внешних команд. Например, можно написать сценарий, запускающий группу программ одновременно или передающий программе сложный набор параметров. Собственно говоря, системные сценарии за пуска, такие как `/etc/rc`, и сценарии регистрации пользователей, такие как `/ shrc` служат лишь для запуска программ, хотя в них и могут использоваться различные вспомогательные программные конструкции

Помимо этого поддерживаются механизмы каналов и переадресации потоков ввода-вывода (см. главу 5), позволяющие формировать цепочки программ. Это дает воз можность написать простой сценарий берущий на себя работу полудюжины команд.

Выполнение внешних команд

Запуск внешней команды организуется чрезвычайно просто: достаточно лишь ука зать ее имя в сценарии, а также список необходимых аргументов Пример простей шего сценария представлен в листинге 31.1.

Листинг 31 1 Минимальный сценарий интерпретатора команд

```
#!/bin/sh
/bin/ls -l
exit 0
```

Если набрать эти строки в файле с именем, скажем, `lsl`, сделать файл исполняе мым и ввести `./lsl`, будут получены те же результаты, что и в случае команды `ls -l` Чтобы такой сценарий было удобно вызывать, скопируйте его в один из каталогов указанных в переменной среды `$PATH`, например `/usr/local/bin`.

В данном примере указывается полный путь к программе `/bin/ls` Это необязательное требование, но так рекомендуется поступать по двум причинам Во-первых, это позволит свести к минимуму риск ис кажения работы сценария, если в одном из программных каталогов или в текущем каталоге появится сценарий с аналогичным именем. Во-вторых, это гарантирует работоспособность сценария даже в том случае, когда пользовательская переменная среды `$PATH` не содержит ссылку на нужный каталог С другой стороны, это уменьшит перено симость сценария, поскольку одна и та же программа в разных систе мах может храниться в разных каталогах

К сожалению, такой сценарий не принимает никаких аргументов. Например, если ввести `./lsl a*`, сценарий отобразит список всех файлов, а не только тех, имена которых начинаются на 'a'. О работе с аргументами командной строки будет рассказываться на с. 613.

Сценарий может вызывать более одной внешней команды. Каждый такой вызов должен стоять в отдельной строке, и сценарий будет обрабатывать их последовательно, дожидаясь завершения каждой программы. Если же в конце вызова стоит оператор `&`, программа запустится в фоновом режиме. Для наглядности рассмотрим листинг 31.2.

Листинг 31.2. Сценарии, запускающии несколько программ

```
#!/bin/sh
/usr/X11R6/bin/xterm &
/usr/X11R6/bin/xclock &
/usr/X11R6/bin/xload &
exit 0
```

Здесь вызываются три программы: `xterm` (терминальное окно), `xclock` и `xload`. Поскольку все это X-программы, они должны выполняться одновременно, отсюда наличие оператора `&` в конце каждой строки. Результатом работы сценария будет появление трех окон на экране.

Каналы и переадресация потоков ввода-вывода

В главе 5 рассказывалось о том, как организовывать каналы и выполнять переадресацию ввода-вывода в командной строке. Все то же самое можно делать и в сценариях интерпретатора команд. Предположим, необходимо написать сценарий, отображающий названия всех сетевых интерфейсов компьютера (листинг 31.3).

Листинг 31.3. Пример формирования канала в сценарии

```
#!/bin/sh
ifconfig | grep flags | cut -f 1 -d " "
exit 0
```

Для наглядности в этом и последующих сценариях не указываются полные пути доступа к внешним программам. При желании можете добавить их самостоятельно. Воспользуйтесь командой `whereis`, чтобы узнать, где находится соответствующий программный файл, например **`whereis ifconfig`**.

В листинге 31.3 выполняются три команды, соединенные каналом. Первая из них — `ifconfig` — выдает информацию о сетевых интерфейсах компьютера, как описывалось в главе 14. Проблема в том, что этой информации слишком много. Рассмотрим типичные результаты работы команды:

```
$ ifconfig
vr0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet 192.168.1.6 netmask 0xffffff00 broadcast 192.168.1.255
        ether 00:80:c8:fa:3b:0a
        media: Ethernet autoselect (100baseTX,
        status: active
lp0: flags=8810<POINTTOPOINT,SIMPLEX,MULTICAST> mtu 1500
lo0  flags=8049<UP,LOOPBACK,MULTICAST> mtu 16384
        inet 127.0.0.1 netmask 0xff000000
ppp0: flags=8010<POINTTOPOINT,MULTICAST> mtu 1500
```

Здесь сообщается о четырех интерфейсах: vr0 lp0 lo0 и ppp0. Лишнюю информацию нужно удалить. Вторая команда — grep — отбирает только те строки, в которых содержатся названия интерфейсов. Во всех этих строках присутствует ключевое слово flags, поиск которого и осуществляет команда grep. Третья команда — cut — удаляет из полученной строки все, кроме первого поля. Опция -f 1 служит указанием на первое поле, а опция -d ":" говорит о том, что разделителем полей является двоеточие. Результат работы сценария будет таким:

```
$ ./ifinfo
vr0
lp0
lo0
ppp0
```

Впоследствии мы усложним сценарий, чтобы решать более интересные задачи. А пока поговорим о том, как сохранить результаты работы сценария в файле. Для этого можно добавить оператор переадресации непосредственно в сценарий, указав необходимый файл:

```
ifconfig | grep flags | cut -f 1 -d " " > /tmp/if-info.txt
```

Полученный файл можно просматривать, редактировать или передавать другим программам для последующей обработки. Аналогичного результата можно добиться с помощью операторов командной строки, заставляя сценарий считывать или записывать файл, имя которого передано ему в качестве аргумента.

Переменные

Переменные — важная часть любого языка программирования, и языки сценариев не являются исключением. Переменные сценариев тесно связаны с переменными среды, о которых рассказывалось в главе 6. Собственно говоря, переменные среды можно интерпретировать в сценариях как обычные переменные. Их имена принято записывать прописными буквами, а значения раскрываются так же, как и в командной строке. Существует лишь несколько важных исключений, связанных с аргументами командной строки.

Присвоение значений переменным

В главе 6 описывалось, как назначить переменной значение в командной строке. Точно так же это делается и в сценариях. В случае интерпретатора Bourne операция присвоения выглядит так:

```
export color=red
```

Здесь переменной $color присваивается значение red. Значением может быть как строка типа red, так и число, например 12. Если строка содержит пробел, ее необходимо взять в кавычки. Знак '$' перед именем переменной указывать не нужно. Он используется лишь при последующих ссылках на переменную.

Желательно отказаться от команды export, поскольку она преобразует обычную переменную в переменную среды. Лучше воспользоваться следующей записью:

```
color=red
```

На заметку Слева и справа от знака равенства не должно быть никаких пробелов. В противном случае интерпретатор либо воспримет имя переменной как команду, либо присвоит ей пустой значение и воспримет последующее значение как команду.

Использование аргументов командной строки в качестве переменных

Переменные широко применяются в сценариях для хранения значений аргументов командной строки сценария Например, сценарию можно передать имя файла или ка талога. Рассмотрим еще раз листинг 31 1. Этот сценарий отображает расширенный список содержимого текущего каталога, и если при вызове сценария указать в качестве аргумента имя другого каталога, оно будет проигнорировано К счастью, существует простой способ восстанавливать значения параметров, переданных сценарию. Каждый такой параметр предетавляется в сценарии нумерованной переменной, от `$0` до `$9` (для доступа к последующим аргументам предназначена команда `shift`). Переменная `$0` содержит имя самого сценария, переменная `$1` — первый аргумент, `$2` — второй и т д. Если нужно узнать, сколько аргументов передано сценарию, воспользуй тесь переменой `$#` В листинге 31.4 показан вариант сценария из листинга 31.1, при нимающий единственный аргумент

Листинг 31.4 Пример передачи параметра сценарию

```
# /bin/sh
ls -l $1
exit 0
```

Нет необходимости явно определять переменные для аргументов, так как это уже сделано в интерпретаторе команд. Если приведенный сценарий называется `lsl` то по команде **`./lsl`** будет выдан расширенный список содержимого текущего каталога, а при указании имени каталога (например, **`./lsl /tmp`**) сценарий отобразит соответст вующий ему список. Одно из ограничений такого подхода, по крайней мере с точки зрения передачи имени файла или каталога сценарию, заключается в обработке мета символов. Интерпретаторы FreeBSD *раскрывают* метасимволы, т.е преобразуют их в список файлов, перед вызовом нужной программы Таким образом если ввести **`./lsl *`**, в переменную `$1` сценария будет записано имя первого файла или подкаталога в текущем каталоге Чтобы отменить предварительную интерпретацию, заключите аргумент с метасимволом в двойные кавычки, например **`./lsl "*"`** Тогда метасимвол '*' будет передан команде `ls` в виде аргумента `$1`, и уже она раскроет его как список содержимого каталога

В качестве аргументов командной строки передаются не только имена файлов На пример, можно написать сценарий, принимающий имя пользователя и обрабатывающий параметры его учетной записи Такой сценарий будет вызывать утилиту `passwd` или `adduser`, передавая ей имя пользователя в переменной `$1`

Применение переменных

В листинге 31 4 иллюстрировалось лишь одно из возможных применений пере менных. они передавались внешним командам Над переменными допускается вы полнять множество различных операций, в частности записывать в них значения, вводимые пользователем

Отображение значения переменной

Для отображения информации в сценариях используется команда `echo` Она пред ставляет собой внешнюю команду, поэтому передача ей переменных осуществляется так же, как и в случае команды `ls` в предыдущем сценарии. Рассмотрим листинг 31.5 в котором осуществляется обработка переменных

Листинг 31 5 Пример манипулирования переменной

```
#!/bin/sh
echo "The input variable is" $1
# Присваиваем значение локальной переменной
myvar=$1
# Выполняем определенные действия
# (будут заданы позднее).
echo "The transformed variable is" $myvar
```

Предположим, этот сценарий называется `scratch`. Его можно вызвать следующим образом:

```
$ ./scratch FreeBSD
The input variable is FreeBSD
The transformed variable is FreeBSD
```

Команда `echo` выводит на экран все переданные ей параметры. Таким образом, она отображает не только строки сообщений, например `The input variable is`, но и значение переменной `$1`. В данном простом сценарии значение этой переменной не модифицируется.

Прием данных от пользователя

В сценариях часто требуется принимать данные от пользователей, например запрашивать регистрационные имена и пароли. Пользователям это удобнее, чем вводить все данные в виде аргументов командной строки. Для чтения данных с консоли предназначена команда `read`. Конец входной строки обозначается нажатием клавиши <Enter>. Применение команды демонстрируется в листинге 31.6.

Листинг 31 6 Прием данных от пользователя

```
#!/bin/sh
echo "Please enter some information
read info
echo "You typed " $info
exit 0
```

Результаты работы сценария примерно таковы:

```
$ ./input-info
Please enter some information
Shakespeare was a great writer of scripts!
You typed Shakespeare was a great writer of scripts!
```

В переменную `info` заносятся любые данные, вводимые пользователем. Это может быть целое предложение, поскольку команда `read` не обращает внимания на пробелы.

Модификация строковых переменных

В широком смысле существуют два типа переменных: *строковые*, содержащие произвольную комбинацию букв и цифр, и *числовые*, содержащие только цифры. Работа с такими переменными ведется по-разному.

Модификация строковой переменной выполняется в рамках операции присваивания. Предположим, необходимо вставить текст в начало и конец строки. Соответствующий сценарий, представляющий собой модификацию листинга 31.5, показан в листинге 31.7.

Листинг 31.7 Добавление данных в строковую переменную

```
#!/bin/sh
echo "The input variable is" $1
# Присваиваем значение локальной переменной
```

```
myvar=$1
myvar="Lions, "$myvar", and bears.
echo "The transformed variable is" $myvar
```

Вот результаты работы сценария

```
$ /add-data tigers
The input variable is tigers
The transformed variable is Lions  tigers, and bears.
```

Следует учитывать, что подобное преобразование неприменимо к аргументным переменным. Вот почему значение переменной `$1` записывалось в локальную переменную `myvar`, над которой уже можно выполнять манипуляции. Как видно из листинга 31.7, чтобы добавить текст в переменную, нужно в правой части оператора присваивания задать ссылку на саму переменную, окружив ее вставляемыми строками. Никаких пробелов перед ссылкой и после нее быть не должно.

Для более сложных манипуляций со строками следует вызывать внешние команды по крайней мере когда речь идет о программировании на языке интерпретатора команд (Perl, Python и некоторые другие языки содержат довольно мощные встроенные средства обработки строк). Рассмотрим некоторые полезные команды:

- **`grep`** — осуществляет поиск текста в файле. Описывалась в главе 8.
- **`cut`** — вырезает строку текста. Операции вырезания можно осуществлять с учетом позиции символа (например, оставить первые 10 символов строки) или на основе символов, гарантированно присутствующих в строке.
- **`sed`** — чрезвычайно мощная команда, позволяющая выполнять все виды текстовых манипуляций.
- **`sort`** — эта команда обычно применяется к целым файлам, осуществляя их сортировку в соответствии с заданным критерием.

Подробнее обо всех этих командах можно прочитать на соответствующих `man` страницах. В простых сценариях достаточно использовать команды `grep` и `cut`, как было показано в листинге 31.3.

Модификация числовых переменных

Над числовыми переменными выполняются математические операции. Например, можно написать короткий сценарий, преобразующий значение емкости диска из гигабайтов, как их понимают изготовители дисков (10^9 байт) в компьютерные гигабайты (2^{30} байт). Сценарий, представленный в листинге 31.8, использует команду `bc`, которая возвращает значение математического или логического выражения.

Листинг 31.8. Сценарий, выполняющий математические вычисления

```
#!/bin/sh
gib= echo "($1 * 1000000000)   1073741824"   bc -l
echo $1 "disk gigabytes is" $gib "computer gigabytes "
exit 0
```

Строка, в которой переменной `$gib` присваивается значение, требует расшифровки. Дело в том, что команда `bc` применяется к файлам, поэтому в листинге 31.8 она является частью канала. Команда `echo` подготавливает значение в том виде, который допустим на входе команды `bc`, и записывает его в канал. Команда `bc` интерпретирует прочитанное из канала значение как файл. Аналогичное выражение нельзя передать ей в виде аргумента командной строки. Вся командная строка, соответствующая каналу, взята в обратные кавычки. Они сообщают интерпретатору команд о том, что заключенный в кавычки текст является командой, результат работы которой должен быть записан в переменную `$gib`. Результат работы самого сценария таков:

```
$ ./gb2gib 65
65 disk gigabytes is 60 535967350006103515б2 computer gigabytes
```

Условные выражения

На момент написания сценария не всегда известно с какими именно входными данными придется иметь дело и нужно ли выполнять те или иные действия. В таких ситуациях следует применять специальные программные конструкции, называемые *условными выражениями*. Они сообщают компьютеру о том, что определенные строки сценария следует выполнять лишь в том случае, когда указанное выражение истинно. С помощью условных выражений можно создавать гораздо более сложные сценарии чем рассмотренные нами до сих пор

Основные логические операции

В основе условных выражений (в частности, инструкций `if` и `elif`) лежат логические проверки. значение выражения должно быть либо истинным, либо ложным. Можно проверять наличие файла, идентичность строк и т.д. В табл. 31 1 приведены условные выражения допустимые в сценариях интерпретатора комаил

Таблица 31.1. Условные выражения и соответствующие логические проверки

Выражение	Условие истинности
`a` *файл*	*Файл* существует
`b` *файл*	Аргумент является файлом блочного устройства
`c` *файл*	Аргумент является файлом символьного устроиства
`d` *файл*	Аргумент является каталогом
`e` *файл*	*Файл* существует
`f` *файл*	Аргумент является обычным фаилом
`g` *файл*	Для *фаила* установлен бит SGID
`h` *файл*	*Файл* является символическои ссылкой
`k` *файл*	Для *фаила* установлен sticky бит
`p` *файл*	Аргумент является именованным каналом
`r` *файл*	Файл доступен для чтения
`s` *файл*	Размер *файла* больше нуля
`t` *дескриптор*	*Дескриптор* фаила открыт и указывает на терминал
`u` *файл*	Для *фаила* установлен бит SUID
`w` *файл*	Файл доступен для записи
`x` *файл*	*Файл* является выполняемым
`O` *файл*	*Файл* принадлежит пользователю, запустившему сценарии
`G` *файл*	*Файл* принадлежит той же группе, что и вольэователь, запустившии сценарий
`L` *файл*	*Файл* является символическои ссылкои
`S` *файл*	Аргумент являются сокетом
`N` *файл*	*Файл* модифицировался с момента последнего чтения
файл1 `-nt` *файл2*	*Файл1* модифицировался позднее, чем *файл2*

Выражение	Условие истинности
файл1 -ot *файл2*	*Файл1* модифицировался раньше, чем *файл2*
файл1 -ef *файл2*	У *файла1* и *файла2* совпадают номера устройств и номера индексных дескрипторов (т. е. они являются жесткими ссылками на один и тот же файл)
-o *опция*	Указанная опция интерпретатора команд установлена
-z *строка*	Длина *строки* равна нулю
-n *строка*	Длина *строки* больше нуля
строка1 = *строка2*	Строки равны
строка1 != *строка2*	Строки не равны
строка1 < *строка2*	*Строка1* сортируется раньше, чем *строка2*
строка1 > *строка2*	*Строка1* сортируется позже, чем *строка2*
аргумент1 -eq *аргумент2*	Целочисленные аргументы равны
аргумент1 -ne *аргумент2*	Целочисленные аргументы не равны
аргумент1 -lt *аргумент2*	Целочисленный *аргумент1* меньше, чем целочисленный *аргумент2*
аргумент1 -le *аргумент2*	Целочисленный *аргумент1* меньше или равен целочисленному *аргументу2*
аргумент1 -gt *аргумент2*	Целочисленный *аргумент1* больше чем целочисленный *аргумент2*
аргумент1 -ge *аргумент2*	Целочисленный *аргумент1* больше или равен целочисленному *аргументу2*

Поскольку сценарии интерпретатора команд часто применяются для обработки файлов, в них предусмотрены многочисленные условные выражения, касающиеся существования файлов и их характеристик, таких как дата модификации, размер и т.д. Независимо от того, относится условное выражение к файлу либо к сравнению строк или чисел, оно должно входить в инструкцию проверки `if` или `elif`.

Инструкции if, elif, else

Двоичные логические проверки (т. е. проверки с двумя состояниями) реализуются с помощью инструкции `if`. В простейшем случае проверка выглядит примерно так

```
if [ $configfile -ot $oldconfigfile ]
then
   echo "Warning!"
   echo "New configuration file appears to be old."
fi
```

В этом фрагменте сравниваются даты модификации двух конфигурационных файлов, имена которых хранятся в переменных `$configfile` и `$oldconfigfile`. Рассмотрим его ключевые особенности

- **Квадратные скобки вокруг условного выражения.** Логическая проверка выполняется в квадратных скобках, стоящих после ключевого слова `if`.
- **Пробелы вокруг условного выражения.** Перед обеими скобками и после открывающей скобки должны стоять пробелы, иначе интерпретатор не сможет правильно распознать выражение.
- **Отступы.** Отступы в третьей и четвертой строках не являются обязательными. В данном случае команды `echo` выполняются только тогда, когда условие истинно, и отступы просто подчеркивают этот факт.

- **Завершающая инструкция** Инструкция `if` завершается ключевым словом `fi` (т.е. слово "if", записанное наоборот). Все что находится между ключевыми словами `then` и `fi`, выполняется, только когда условное выражение истинно

Если необходимо задать дополнительные условия воспользуйтесь вложенной инструкцией `elif`. Например, после сравнения дат модификации можно проварить наличие третьего файла. Инструкция `elif` имеет ту же структуру что и `if` только завершающее ключевое слово `fi` у них общее

Можно задавать иерархию проверок `elif` в рамках единой инструкции `if` Можно также предусмотреть программный код на тот случай если все условия окажутся ложными Этот код задается во вложенной инструкции `else`. Команды, расположенные между ключевыми словами `else` и `fi`, выполняются только тогда, когда все выше стоящие проверки не дали результата

Чтобы проиллюстрировать действие инструкций `if elif` и `else`, рассмотрим листинг 31 9 Этот сценарий идентифицирует набор микросхем сетевой платы Ethernet на основании имени связанного с ней устройства, сообщаемого командой `ifconfig`. Для достижения поставленной цели сценарий сначала выделяет имена всех сетевых интерфейсов с помощью команд `ifconfig`, `grep` и `cut`, как это делалось в листинге 31 3. Полученная переменная `$iface` будет содержать имена всех интерфейсов, поэтому с помощью команд `echo` и `cut` в сценарии изолируется имя первого интерфейса. Теперь переменная `$iface` содержит строку, значение которой можно проверить в инструкции `if`. В данном случае порядок проверок не важен. В основной инструкции значение переменной сравнивается со строкой `fxp0`, которая идентифицирует плату Intel EtherExpress Pro 100B. Если проверка прошла успешно, сценарий выводит соответствующее сообщение и пропускает остальные проверки В противном случае в инструкции `elif` выполняется следующее сравнение, позволяющее выявить плату DEC/Intel Tulip. Вторая проверка `elif` идентифицирует плату VIA Rhine. Если ни одна из проверок не дала результата, сценарий переходит к инструкции `else`, в которой пользователю сообщается о том, что идентифицировать устройство не удалось и приводится имя устройства

Листинг 31 9 Применение инструкции `if`, `elif` и `else`

```
#'/bin/sh

# Получение имен всех интерфейсов.
iface= ifconfig | grep flags   cut  f 1 -d ":"

# Выделение имени первого интерфейса (обычно это первое
# устройство Ethernet).
iface= echo $iface | cut -f 1 -d " "

# Идентификация устройства
if [ $iface = "fxp0" ]
then
   echo "This computer has an Intel EtherExpress board "
elif [ $iface = "de0" ]
then
   echo "This computer has a DEC/Intel Tulip chipset "
elif [ $iface = "vr0" ]
then
   echo "This computer has a VIA Rhine chipset "
else
   echo "I can't identify this computer's primary network card
   echo "The unidentified interface name is" $iface
fi
exit 0
```

На заметку Для наглядности в листинге 31 9 приведены только три сравнения В полном варианте сценария были бы десятки проверок, по одной для каждого драйвера Ethernet во FreeBSD

Приведенный сценарий либо распознает используемый набор микросхем Ethernet, либо сообщит о невозможности сделать это. К примеру, результат работы сценария может быть таким

```
$ ./identify
This computer has a VIA Rhine chipset.
```

Сценарий в листинге 31 9 можно модифицировать различными способами. Если допустим, нужно идентифицировать только один набор микросхем, удалите обе инструкции `elif`. Если тот факт, что распознать интерфейс не удалось, не играет роли удалите инструкцию `else`. Можно добавлять различные команды оптимизирующие сетевые настройки в зависимости от имеющегося оборудования

Инструкция case

Инструкции `if` и `else` удобны для выполнения двоичных проверок, когда условие либо истинно, либо нет. Вложенная инструкция `elif` позволяет учитывать дополнительные условия. Например, в листинге 31.9 выявлялись три конкретных набора микросхем Ethernet, а инструкция `else` содержала код для остальных случаев. Такая процедура становится слишком неудобной, когда число возможных условий велико На этот случай предусмотрена еще одна условная инструкция. `case`. В ней выполняется один из блоков команд в зависимости от значения проверяемой переменной Формат инструкции таков:

```
case переменная in
    значение1) блок_команд1
               ;;
    значение2) блок команд2
               ;;
        |
esac
```

После ключевого слова `case` указывается имя переменной, включая начальный символ '$'. Далее проверяются ее возможные значения Если обнаруживается совпадение, выполняются соответствующие команды, а остальные проверки пропускаются. Каждый блок команд завершается оператором `;;`. Количество проверок может быть произвольным Некоторые проверки имеют особое значение. К примеру, допускается использование метасимвола '*', соответствующего произвольному значению Как правило, такая проверка осуществляется в самом конце и задает действия, выполняемые по умолчанию если ни одна из проверок не дала результата Это можно считать аналогом инструкции `else` Сама инструкция `case` завершается ключевым словом `esac` Рассмотрим ее применение на примере листинга 31 10 который функционально эквивалентен листингу 31 9.

Листинг 31 10 Применение инструкции case

```
#!/bin/sh

# Получение имен всех интерфейсов.
iface=`ifconfig | grep flags | cut -f 1  d " "

# Выделение имени первого интерфейса.
iface= echo $iface   cut -f 1 -d " "

# Идентификация устройства
```

```
case $iface in
   fxp0) echo "This computer has an Intel board."
         ;;
   de0) echo "This computer has a DEC/Intel Tulip chipset "
        ;;
   vr0) echo "This computer has a VIA Rhine chipset."
        ;;
   *) echo " I can't identify this computer's primary network card "
      echo  The unidentified interface name is" $iface
      ;;
esac
exit 0
```

Инструкцию case проще вводить, чем if/elif, поскольку не нужно многократно указывать имя переменной, квадратные скобки и знаки равенства С другой стороны в инструкциях if/elif удобнее выполнять различные проверки, связанные со статусом файла (существует ли он когда последний раз модифицировался и т.д.).

Условное выполнение команд

В сценариях часто используется удобная возможность условного выполнения команд. Например:

```
tar cvz payroll-backup tgz /home/payroll/ && rm  r /home/payroll/*
```

Вспомните, что многие программы и сценарии возвращают код завершения 0 в случае успешного завершения и другое число при возникновении ошибки. Оператор && стоящий между двумя командами, заставляет интерпретатор проверить код завершения первой команды и выполнить вторую команду только в том случае, когда первая команда возвращает нулевой код Таким образом, в предыдущем примере файлы из каталога /home/payroll удаляются тишь при условии, что предыдущая команда tar успешно заархивировала каталог Если команда tar сталкивается с проблемами каталог не удаляется

Родственный оператор означает, что вторая команда выполняется только в том случае, когда первая команда терпит неудачу Его можно использовать для вывода сообщений об ошибках

Циклы

Важной группой конструкций сценариев являются *циклы*. Они позволяют сценариям выполнять одни и те же действия многократно, например последовательно воспроизводить все звуковые файлы каталога или обновлять параметры каждой учетной записи. Существует несколько типов циклов, различающихся условиями останова (т е обстоятельствами при которых происходит выход из цикла)

Циклы for

В цикле for команды выполняются фиксированное число раз: по одному разу для каждого аргумента в списке проверки Этот список может представлять собой строку состоящую из имен файлов, устройств пользователей и т.д Но часто список формируется непосредственно в инструкции for путем заключения команды создающей список, в обратные кавычки Формат инструкции for таков

```
for переменная in список
do
    команды
done
```

Переменная цикла может объявляться непосредственно в инструкции `for`. Если она использовалась ранее, ее значение переопределяется каждым элементом списка. Между ключевыми словами `do` и `done` могут стоять любые команды. В качестве иллюстрации рассмотрим листинг 31.11, представляющий собой модификацию листинга 31.10. Предыдущий сценарий идентифицировал только первый интерфейс, который обычно является интерфейсом Ethernet, хотя это не обязательно. Новый сценарий последовательно анализирует каждый элемент в списке сетевых интерфейсов, что позволяет ему распознавать не только Ethernet-устройства, но и PPP-интерфейсы, а также интерфейс обратной связи. Поскольку в цикле `for` каждый элемент списка автоматически заносится в переменную `$iface`, нет необходимости задавать команду, выделяющую имя очередного интерфейса. Вместо этого команда, генерирующая список интерфейсов, встроена в саму инструкцию `for`.

Листинг 31.11. Применение инструкции `for`

```
# /bin/sh

for iface in `ifconfig | grep flags | cut -f 1 -d ":"`
do
   case $iface in
      lo0) echo "This computer has a loopback interface"
           ;;
      ixp0 ) echo "This computer has an Intel board "
           ;;
      de0) echo "This computer has a DEC/Intel Tulip chipset "
           ;;
      vr0) echo "This computer has a VIA Rhine chipset"
           ;;
      ppp0 ) echo "This computer has a PPP interface."
           ;;
       ) echo " I can't identify an interface " $iface
         ;;
   esac
done
exit 0
```

Этот сценарий генерирует по одному сообщению для каждого сетевого интерфейса. Результат будет примерно таким:

```
$ ./identify3
This computer has a VIA Rhine chipset
I can't identify an interface. lp0
This computer has a loopback interface
This computer has a PPP interface.
```

Как видим, сценарий распознал три из четырех сетевых интерфейсов компьютера.

Циклы while

Второй тип циклов создается с помощью инструкции `while`. Этот цикл выполняется до тех пор, пока заданное условие истинно. Формат инструкции таков:

```
while [ условие ]
do
   команды
done
```

Такого рода структуру можно применять для выполнения операций все то время, пока файл существует, или две переменные равны друг другу, или пользователь не решит прервать цикл. Применение цикла `while` иллюстрируется в листинге 31.12. Этот сценарий вызывает утилиту `passwd` для указываемых пользователей, позволяя последовательно менять их пароли.

Листинг 31.12. Применение цикла `while`

```
#!/bin/sh
echo "This script changes multiple user passwords "
echo "Enter a username ('n  to exit):"
read username
while [ $username != "n" ]
do
    passwd $username
    echo "Enter a username ( n  to exit) "
    read username
done
exit 0
```

Сценарий, показанный в листинге 31.12, должен запускаться суперпользователем, поскольку лишь ему разрешено менять пароли других пользователей.

Сценарий запрашивает имя пользователя после чего вызывает утилиту `passwd` для этого пользователя В свою очередь, утилита `passwd` дважды запрашивает новый пароль (второй раз — для подтверждения) Результаты работы сценария выглядят примерно так:

```
# ./multipasswd
This script changes multiple user passwords
Enter a username ('n' to exit):
tennyson
Changing local password for tennyson
New password.
Retype new password:
passwd: updating the database...
passwd: done
Enter a username ( n  to exit):
wilder
Changing local password for wilder
New password.
Retype new password:
passwd: updating the database...
passwd: done
Enter a username   n  to exit):
n
```

Следует отметить, что проверяемая переменная `$username` должна быть проинициализирована до начала цикла. Из-за этого в цикле `while` нередко дублируется код связанный с обновлением переменной цикла. В листинге 31.12 имя первого пользователя запрашивается перед началом цикла, а имена остальных пользователей — уже в самом цикле

Циклы until

Третий тип циклов создается с помощью инструкции `until`. Этот цикл выполняется до тех пор пока заданное условие не станет истинным Формат инструкции таков

```
until [ условие ]
do
    команды
done
```

Инструкции `until` и `while` настолько похожи, что могут быть взаимозаменяемы Например, в листинге 31.12 достаточно заменить ключевое слово `while` на `until` и в качестве условия цикла указать `$username != "n"` В листинге 31.13 сделано чуть больше изменений. Чтобы не дублировать запрос имени пользователя вне цикла пе

ренесем его в начало цикла и поместим вызов утилиты passwd в инструкцию if во избежание обработки несуществующей учетной записи n. Кроме того, перед началом цикла переменная $username инициализируется фиктивным значением scratch (такой же подход можно применять в цикле while) В случае листингов 31 12 и 31.13 эти изменения ничего не дают в плане эффективности сценария, лишь немного увеличивают его размер. Но если бы для создания проверяемой переменной требовалось несколько строк, имело бы смысл включить соответствующий код только в цикл

Листинг 31.13 Применение инструкции `until`

```
#!/bin/sh
echo "This script changes multiple user passwords
username="scratch"
until [ $username != "n  ]
do
    echo "Enter a username ( n  to exit) "
    read username
    if [ $username '= "n" ]
    then
        passwd $username
    fi
done
exit 0
```

Функции

Последней важной конструкцией сценариев являются *функции*. Это подпрограммы, вызываемые из других частей программы. Например, вместо того чтобы выполнять громоздкую операцию в одном большом программном блоке, можно разбить его на несколько логических фрагментов, поместить каждый из них в отдельную функцию и последовательно вызвать все эти функции. Рассмотрим, каковы преимущества функционального подхода в сравнении с линейным способом программирования

- **Смысловое упорядочение операций**. Человеку всегда трудно понять назначение очень большого фрагмента кода Если же разбить его на несколько небольших фрагментов, проследить работу программы будет гораздо проще.
- **Наглядность**. Бывает, что фрагмент кода сложно воспринимать, из за того что он вложен в многочисленные циклы и условные конструкции, или же часто встречающаяся процедура разделяет строки, тесно связанные между собой. Перемещение вложенных или повторяющихся фрагментов в отдельные функции способствует повышению удобочитаемости программы в целом.
- **Упрощение отладки**. Разделение программы на функциональные блоки упрощает процесс отладки Можно написать программу "заглушку", которая будет вызывать функции по отдельности для целей тестирования
- **Устранение нежелательного дублирование кода**. Иногда приходится выполнять одни и те же действия в разных местах программы Вместо того чтобы дублировать соответствующий код, можно поместить его в функцию, заменив в теле программы вызовом функции. Это уменьшит размер программы и число потенциальных ошибок в ней Кроме того, проще вносить изменения в одну функцию, чем выискивать одинаковые фрагменты кода по всей программе.

Базовый синтаксис функции таков

```
имя_функции ()
    команды
}
```

имя_функции

Определение функции начинается с ее имени, после которого стоят круглые скобки пробел и открывающая фигурная скобка. Тело функции состоит из одной или нескольких команд, заключенных между открывающей и закрывающей фигурными скобками. Для вызова функции, как и внешней программы, достаточно указать ее имя

Применение функций иллюстрируется в листинге 31.14. Это модифицированный вариант листинга 31 11, в котором инструкция `case` находилась в цикле `for`. И хотя эта инструкция не была слишком глубоко вложенной, в более сложных случаях такой подход позволяет уменьшить степень вложенности и повысить наглядность сценария. Заметьте также, что в листинге 31 14 функция обращается к переменной `$iface`, определенной в основном теле сценария

Листинг 31.14 Применение функции

```
#'/bin/sh

identify ) {
   case $iface in
      lo0) echo "This computer has a loopback interface "
         ;;
      fxp0) echo "This computer has an Intel board
         ;;
      de0, echo "This computer has a DEC/Intel Tulip chipset "
         ;;
      vr0, echo "This computer has a VIA Rhine chipset."
         ;;
      ppp0) echo "This computer has a PPP  nterface."
         ;;
      *) echo "I can't identify an interface:" $iface
         ;;
   esac
}

for iface in  ifconf g   grep flags   cut  f 1  d   "
do
   identify
done
exit 0
```

Когда возникает необходимость в функциях? Как правило, они используются в сценариях, занимающих несколько экранов строк и содержащих множество условных конструкций и циклов, четкую структуру которых сложно продумать наперед. Например, если необходимо написать сценарий, предлагающий пользователю множество вариантов выбора, создайте для каждого варианта отдельную функцию, а затем вызывайте эти функции в инструкции `case` Другой распространенный подход заключается в следующем Если сценарий не помещается на экране, его разбивают на несколько функций, каждая из которых занимает в точности один экран. Не строит впрочем слепо придерживаться такого подхода в случаях, когда функция выполняет хоть и громоздкую, но линейную последовательность действий, таких как отображение множества строк текста, инициализация множества переменных и т д

Резюме

Умение писать сценарии очень важно для системного администратора FreeBSD, да и для рядовых пользователей это не будет лишним. Сценарии позволяют автоматизировать выполнение рутинных задач, объединяя в единое целое несвязанные программы. Сценарии могут также содержать различные управляющие конструкции, связанные с принятием решений, многократным повторением одних и тех же действий и т.д. Существуют различные языки написания сценариев, у каждого из которых есть сильные стороны и круг преданных сторонников. Для системных администраторов наиболее важны языки интерпретаторов команд, поскольку именно на них пишутся сценарии запуска системы. Таким образом, знание этих языков позволит читателям лучше понимать, как работают важные системные сценарии и как их модифицировать.

Независимо от используемого языка нужно понимать ключевые конструкции всех сценариев: вызовы внешних программ, переменные, условные выражения, циклы и функции. С их помощью можно писать довольно сложные сценарии даже на простом языке интерпретатора команд.

Полный справочник по

Глава 32

Поиск и устранение неполадок

В этой книге рассказывается о том как работает FreeBSD, — точнее, как она *должна* работать. В каждой главе указывалось на возможность возникновения тех или иных проблем, однако все проблемы предсказать невозможно. Даже если бы это было так, их описание удвоило бы размер книги. В настоящей главе будут рассмотрены общие подходы к поиску и устранению различных неполадок в системе. Сначала мы поговорим о том, где искать необходимую информацию и куда можно обращаться за помощью. Затем будет указано, как определить, к аппаратной или программной области относится неполадка. Далее следует локализовать неполадку, а для этого необходимо знать, какие проблемы наиболее часто возникают в системе. Следующий шаг заключается в тестировании сделанных изменений. Наконец, в завершение главы читатели познакомятся с системами аварийной загрузки, которые играют важную роль в устранении серьезных проблем, мешающих нормальной загрузке системы.

Получение справки

Прежде чем приступать к устранению неполадок, полезно узнать о том, где можно найти информационную поддержку и получить техническую помощь. Мы рассмотрим три категории ресурсов: встроенные (доступные во FreeBSD) печатные и интерактивные (доступные в Internet). У каждой из этих категорий свои достоинства, которые играют важную роль при решении определенных проблем. Не стоит также забывать о персональной помощи со стороны друзей, коллег-специалистов и даже платных консультантов. В процессе длительной эксплуатации системы пользователям наверняка предстоит прибегнуть к такой помощи.

Справочные ресурсы содержат информацию о системе, поэтому они полезны для выяснения того, как она *должна* работать (некоторые "проблемы" являются следствием элементарного непонимания функциональных особенностей системы). В ряде источников описываются основные симптомы неполадок и объясняются вероятные причины их возникновения, т.е. они чрезвычайно важны для диагностирования неполадок. Другие ресурсы помогают понять, как устранить возникшую неполадку.

Справочные ресурсы FreeBSD

Прежде всего следует упомянуть справочную информацию, имеющуюся в свыои FreeBSD. В основном это документация различных форматов и разной степени детализации. Рассмотрим наиболее важные виды документации.

- **Комментарии к выпуску**. Если FreeBSD инсталлируется с компакт диска, поищите на нем файлы, имена которых записаны прописными буквами с расширением `.TXT` или `.HTM`. В частности, должны присутствовать следующие текстовые файлы: `ERRATA.TXT`, `HARDWARE.TXT`, `INSTALL.TXT`, `README.TXT` и `RELNOTES.TXT`. В них приведена общая информация о FreeBSD. Файл `HARDWARE.TXT` содержит список поддерживаемых аппаратных устройств, а файл `INSTALL.TXT` поможет справиться с проблемами, возникающими в процессе инсталляции. Все эти файлы доступны также на Web-узле FreeBSD.
- **Системная документация**. В каталоге `/usr/share/doc` содержится большой объем документации по FreeBSD. Документация на английском языке доступна в каталоге `/usr/share/doc/en`. В нем есть два подкаталога `articles` и `books`, в каждом из которых несколько подкаталогов. Подкаталог `articles` содержит статьи размером несколько печатных страниц, касающиеся решения

конкретных задач, таких как конфигурирование мультизагрузочной системы и применения определенных средств, например Zip-дисков. В подкаталоге `books` хранятся более крупные документы, в частности FreeBSD Handbook и FreeBSD FAQ. Все статьи и книги доступны в текстовом виде, в формате HTML, иногда в формате PostScript. Если выполняется обновление системы, загрузите обновленные файлы документации с помощью утилиты `cvsup` а затем введите **`make install`** в каталоге `/usr/share/doc`

Создайте закладки" в броузере, ссылающиеся на адреса `file:///usr/share/doc/en/books/` и `file:///usr/share/doc/en/articles/` Это позволит получать быстрый доступ к спискам книг и статей

- **Документация к пакетам** Многие программы из коллекций пакетов и переносимых приложений FreeBSD поставляются с файлами документации Большинство таких файлов находится в каталоге `/usr/local/share/doc` Формат файлов документации зависит от пакета Это могут быть текстовые файлы HTML-файлы, PostScript-файлы и др В одних пакетах документация представлена в минимальном объеме — к примеру, один лишь файл лицензионного соглашения В других она сопоставима с объемом книги
- **Страницы интерактивного руководства** По традиции основная утилита работы с документацией во FreeBSD называется `man` (сокращение от "manual" — руководство) Файлы документации, представленные в формате этой утилиты, называются *man-страницами* Такие файлы есть у большинства команд, конфигурационных файлов и программ Чтобы вызвать нужную страницу, введите **`man`** *тема*, где *тема* — это название команды, файла (без указания пути) или рубрики. Страницы интерактивного руководства написаны в довольно сжатом стиляе Они больше полезны как справочные, а не учебные материалы
- **Информационные страницы.** В последние годы утилита `info` стала дополнением, а в некоторых случаях — заменой утилите `man` в качестве средства ведения документации. Работа с ними ведется практически одинаково, и выдаваемая информация во многих случаях тоже идентична. Утилита `tkinfo` является графической надстройкой к `info` Сходная надстроика реализована и в редакторе Emacs
- **Программная справка** Многие программы особенно графические, располагают средствами вызова справки обычно доступными в меню Help. Справочная информация может отображаться в небольшом окне справки или же в Web-броузере

Не стоит недооценивать справочные ресурсы FreeBSD. Их не всегда легко найти тем кто мало знаком с FreeBSD (и UNIX вообще), но сама документация часто оказывается чрезвычайно полезной. Особенно хорошо, если сравнивать с другими операционными системами, написана системная документация, находяшаяся в каталоге `/usr/share/doc`. Для пользователей FreeBSD man- и `info`-страницы очень удобны в том плане, что они позволяют быстро вспомнить синтаксис нужной команды или конфигурационного файла Правда, по большей части эти ресурсы не содержат сведений об устранении неполадок. В них, как правило, даются лишь минимальные предположения о том, что может пойти не так зато приводятся подробные сведения о том, как применять различные программы

Печатная документация

Печатные материалы содержат ту же информацию, что и файлы документации, поставляемые вместе с FreeBSD или публикуемые в Internet, с той лишь разницей, что эти сведения не обновляются С другой стороны, у книг и брошюр свои преиму-

шества. Они всегда под рукой, у них прекрасные оглавления и предметные указатели, упрощающие поиск необходимой информации. Можно зайти в книжный магазин и полистать книги по нужной тематике, выбрав те из них, которые подойдут по охвату материала и цене. А самое главное, они доступны даже тогда, когда компьютер не функционирует и другого в вашем распоряжении нет.

Помимо книг есть различные журналы по соответствующей тематике. Наиболее важные из них — *Daemon News* (`www.daemonnews.org`) и *Sys Admin Magazine* (`www.sysadminmag.com`). Первый из них посвящен операционным системам семейства BSD, включая FreeBSD, а второй — системному администрированию в целом, хоть и с упором на UNIX-системы. В обоих журналах приводится ценная информация, касающаяся поиска и устранения неполадок. Важные сведения есть также на Web-узлах этих журналов, включая статьи из старых номеров.

Интерактивные источники информации

Если система работает достаточно надежно для того, чтобы предоставлять доступ в Internet, или этот доступ можно получить с другого компьютера, воспользуйтесь источниками информации, которые есть в глобальной сети. Во многих случаях эти источники в буквальном смысле интерактивны: можно задать вопрос и получить ответ от реального человека, причем довольно быстро. Другие источники нередко дублируют документацию, имеющуюся во FreeBSD, хотя содержащаяся в них информация может быть более свежей. Рассмотрим наиболее важные интерактивные источники.

- **Web-узел FreeBSD.** Основной Web-узел FreeBSD (`www.freebsd.org`) — хороший отправной пункт для получения информации о системе. На этом Web-узле есть та же документация, что и в самой системе. Особенно важным ресурсом является страничка FreeBSD Security Information (`http://www.freebsd.org/security/`). Там можно найти инструкции по восстановлению системы в случае сбоя (в главе 29 рассказывалось о том, что следует делать, если выявлен факт взлома системы).
- **Вспомогательные Web-узлы.** Некоторые Web-узлы посвящены FreeBSD, хотя и не являются официальными. Они тоже содержат много полезной информации. В качестве примеров назовем BSD Vault (`www.bsdvault.net`), Fresh Ports (`www.freshports.org`) и FreeBSD Diary (`www.freebsddiary.org`).
- **Web-узлы, посвященные вопросам безопасности.** К этой категории относится большое число Web-узлов. Наиболее важные из них — CIAC (Computer Incident Advisory Capability — Консультативная служба компьютерных сбоев) по адресу `http://www.ciac.org/ciac/` и CERT/CC (Computer Emergency Response Team Coordination Center — координационный центр Группы компьютерной "скорой помощи") по адресу `www.cert.org`.
- **Группы новостей Usenet.** Это глобальные дискуссионные форумы, открытые для всех желающих. Для чтения новостей необходима специальная программа, такая как `tin`, KNode или Pan (все они есть во FreeBSD). Нужно также знать доменное имя сервера новостей. Связавшись с сервером, программа чтения новостей сделает доступными различные телеконференции, которые организованы в иерархическую систему. Для пользователей FreeBSD особый интерес представляют конференции раздела `comp.unix.bsd.freebsd`. Существуют также форумы, посвященные конкретным программам или темам, например `comp.security.ssh` и `comp.periphs.printers`. Прежде чем задавать вопрос в форуме, попробуйте поискать ответ на него в архивах новостей Google Groups Usenet (`http://groups.google.com`). Если поиски окажутся безрезультатными, обратитесь в форум. Ответ может быть получен через несколько

минут или часов. Хотя конечно, гарантии нет никакой, ведь группы новостей — это своего рода клубы по интересам, там нет людей, которым платят за консультации. Тем не менее это очень важный источник получения помощи.

- **Списки рассылки.** Списки рассылки функционируют примерно так же как и группы новостей, только в качестве среды доставки используется электронная почта, а не протоколы новостей Usenet. Чтобы подписаться на рассылку, нужно отправить письмо по определенному адресу. С этого момента подписавшемуся начнут приходить письма, адресованные всем участникам рассылки, и он сможет сам посылать письма. Перечень списков рассылки FreeBSD, а также инструкции о том, как подписаться на них, доступны по адресу `http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/eresources.html#ERESOURCES-MAIL`. Для новичков наиболее полезным будет список `freebsd-questions`, хотя через него проходит огромное число сообщений. Списков рассылки, посвященных FreeBSD, больше, чем групп новостей, и в них вовлечено большое число людей, так что эти списки являются чрезвычайно ценными источниками информации, касающейся устранения неполадок в системе. Разумеется, общаясь по электронной почте с незнакомыми людьми, следует придерживаться правил хорошего тона. В частности, прежде чем обратиться с вопросом в список рассылки, поищите ответ на него в архивах.
- **Чат.** IRC (Internet Relay Chat) — это виртуальная среда общения пользователей глобальной сети в реальном времени. Ее можно считать более непосредственным вариантом форумов Usenet, а посему — ценным средством оперативного решения проблем. Для общения в чате необходима программа клиент, например `ircii` или `xchat` (обе они доступны в разделе `irc` коллекций пакетов и переносимых приложений FreeBSD). Далее нужно выбрать сеть IRC и подключиться к одному из ее узлов. Информация о сетях IRC доступна по адресу `http://www.irchelp.org/irchelp/networks/`. Потребуется также найти соответствующий *канал* в пределах сети. Это аналог форума Usenet, только каналы не имеют иерархической организации.

Интерактивные источники информации являются хорошим средством получения индивидуальной помощи от других людей. На Web-узлах публикуется статическая документация, тогда как группы новостей, списки рассылки и IRC-каналы позволяют любому желающему связываться с тысячами пользователей FreeBSD. Коллективный опыт сообщества пользователей может оказать неоценимую помощь в решении проблем. Но не забывайте вносить и свой вклад в общее дело. Если у вас есть определенный опыт работы в системе, не оставляйте без внимания просьбы тех пользователей, которым вы можете помочь. Именно из таких "кирпичиков" и складывается глобальная среда.

Выявление аппаратных и программных проблем

Первый этап диагностики проблемы заключается в том, чтобы определить, где находится ее источник, в аппаратной или программной части. Иногда это трудно сделать, поскольку у различных проблем могут быть схожие проявления. Особенно трудно отличить аппаратные проблемы от сбоев драйверов, работающих на чрезвычайно низком уровне. К счастью, проблемы с драйверами редко возникают во FreeBSD и более характерны для новых устройств, драйверы которых еще не были в достаточной степени отлажены.

Распространенные признаки аппаратных проблем

Аппаратные проблемы сложнее всего выявлять и исправлять. Они могут проявляться спорадически а решением часто является замена аппаратных компонентов Для крупной организации, где есть много резервных компонентов и даже свободный компьютер, из которого можно временно позаимствовать нужное устройство, такой подход вполне приемлем Для остальных же это слишком дорогое удовольствие[1]

Итак, в первую очередь необходимо определить, действительно ли имеет место аппаратная проблема. Рассмотрим типичные признаки таких проблем

- **Непредсказуемое функционирование** Некоторые аппаратные проблемы проявляются случайным образом внезапная перезагрузка системы, внезапный сбой видеосистемы и т.д Программные проблемы обычно более предсказуемы хотя это нельзя считать универсальным правилом
- **Аварийные сбои системы.** FreeBSD имеет репутацию высоконадежной операционной системы, и аварийные сбои в ней возникают редко Если компьютер сбоит или самопроизвольно перезагружается слишком часто (хотя бы несколько раз в год), можно предположить именно аппаратную проблему Другая возможная причина – неправильный драйвер, который можно исправить путем обновления ядра
- **Неполадки, касающиеся аппаратных устройств, а не программ.** С аппаратными устройствами работает множество программ, поэтому аппаратные проблемы вызывают сбои одновременно большого числа программ. Например, вышедшая из строя сетевая плата станет причиной неполадок Web-броузеров, почтовых клиентов и других сетевых программ.
- **Невозможность загрузить компьютер** Серьезные аппаратные проблемы иногда препятствуют загрузке компьютера К такому же результату могут привести и ошибки в конфигурации системы, но симптомы таких ошибок будут несколько иными. Как правило, если "виноват" аппаратное устройство, процесс загрузки остановится до появления экрана загрузчика FreeBSD или в процессе отображения сообщений о загрузке ядра Если же вывод сообщений ядра завершен и система хотя бы начала отображать сообщения, связанные со сценарием `/etc/rc`, значит это не аппаратная проблема
- **Неполадка, возникающая только на этом компьютере** Если FreeBSD инсталлирована на многих компьютерах и везде сконфигурирована идентично, то проблема, проявляющаяся только в одной из систем, вероятно, является аппаратной Но быть полностью в этом уверенным нельзя, потому что причина может заключаться в конфигурационной ошибке, действительно свойственной какому то одному компьютеру
- **Неполадки, возникающие во всех операционных системах компьютера.** Если на компьютере помимо FreeBSD инсталлирована другая операционная система, проверьте ее функционирование. Проблема, проявляющаяся и в той, и в другой среде, вероятнее всего, имеет аппаратную природу, иначе можно предположить программную ошибку (иногда драйверы одной из операционных систем не сигнализируют о проблеме, которая вообще говоря, является аппаратной)

[1] Например, в процессе работы над этой книгой материнская плата одного из моих компьютеров вышла из строя, но я не смог определить, что процессор или материнская плата — стало источником проблем, пока не поменял процессор В результате пришлось менять и процессор, и материнскую плату, и еще оперативную память, поскольку для новой материнской платы требовались иные микросхемы памяти

Если компьютер не является мультизагрузочным, попробуйте запустить демонстрационную версию другой операционной системы с компакт диска. Для этой цели могут подойти демонстрационные версии Linux, доступные на Web-узле `www.linuxiso.org`, в частности DemoLinux и SuSE Live Evaluation. Упомянем также FreeDOS (`www.freedos.org`) — открытую реализацию старой операционной системы DOS, которая может загружаться и работать с дискеты.

Следует понимать, что все это — лишь *типичные* аппаратные проблемы. На самом деле их гораздо больше. Некоторые проблемы вообще не проявляются ни одним из указанных симптомов. В то же время, чем больше таких симптомов, тем вероятнее, что имеет место аппаратная проблема.

Распространенные признаки программных проблем

Проблемы программного характера существуют везде, даже во FreeBSD. Они могут быть следствием ошибок в программном коде или неправильной конфигурации и приводить к неадекватной (например, слишком медленной) работе программ. Некоторые из таких проблем непросто решить, поскольку они были "заложены" на этапе разработки программы и являются принципиальными. Не так-то легко, допустим, повысить производительность программы, если в ней используется неэффективный алгоритм (эту проблему можно решить за счет более быстрого процессора). Как правило, устранение программных неполадок сводится к обновлению программы или изменению конфигурации системы.

Программные проблемы проявляются самыми разными симптомами. Рассмотрим наиболее распространенные из них.

- **Повторяющиеся неполадки.** Программные проблемы часто проявляются в конкретных ситуациях, когда, например, загрузка определенного файла вызывает сбой текстового процессора или при определенной конфигурации сервер слишком рано отключается от клиента.
- **Неполадки, относящиеся к программам, а не к аппаратным устройствам.** Большинство рассматриваемых проблем связано с конкретными программами или логическими подсистемами FreeBSD. Если, например, определенный звуковой редактор не работает должным образом, тогда как остальные звуковые утилиты функционируют нормально, то проблема — в редакторе, а не в звуковой плате.
- **Неполадки, возникающие на многих компьютерах.** Если есть несколько одинаково сконфигурированных FreeBSD-систем, то программная проблема должна проявляться во всех этих системах.
- **Неполадки, локализованные во FreeBSD.** Если на компьютере помимо FreeBSD инсталлирована другая операционная система и в ней анализируемая проблема не возникает, то она, скорее всего, связана с программным обеспечением, драйверами или конфигурацией FreeBSD.
- **Неполадки, локализованные в одной программе или подсистеме.** Многие проблемы возникают из-за ошибок отдельных программ и соответственно, проявляются только в этих программах. Иногда проблема затрагивает группу взаимосвязанных программ, поскольку все они используют одну и ту же библиотеку или имеют общую конфигурацию. Например, неправильная конфигурация сетевого интерфейса отразится на всех сетевых программах. Впрочем, ошибка может содержаться в настолько важной подсистеме, что это вызовет сбои в работе большого числа программ и существенно затруднит диагностику.

- **Неполадки, с которыми сталкивается один пользователь или группа пользователей.** Иногда проблемы возникают только у некоторых пользователей. Такие проблемы могут быть связаны с особенностями конфигурации учетной записи, а также правами доступа к каталогам, программам или файлам устройств. Если пользовательские конфигурационные файлы повреждены, создайте тестовую учетную запись с корректными конфигурационными файлами, чтобы воспроизвести проблемную ситуацию.

Как и в случае аппаратных проблем, это далеко не полный список. Возникшая проблема может не соответствовать ни одному из перечисленных признаков, поэтому универсального рецепта все равно не существует.

Локализация проблемы

Диагностика проблем, как аппаратных, так и программных, подразумевает постепенное сужение круга возможных причин. К любой неполадке можно применить общие приемы решения проблем, но рано или поздно придется вести анализ в конкретной предметной области. Например, можно не знать о том, что близко расположенные люминесцентные лампы и источники сильных электромагнитных полей вызывают мерцание изображения на экране монитора, и, соответственно, не суметь устранить эту проблему. Объем знаний, необходимых для полноценной диагностики возможных проблем, огромен, поэтому в данном разделе мы дадим лишь общие советы и подсказки по поводу локализации неполадок.

Общие процедуры локализации

Процесс поиска и устранения компьютерных неполадок напоминает научное исследование. Ученый разрабатывает эксперимент для проверки двух или более возможных гипотез. Каждая из них даст различные предсказания, и эксперимент позволяет их проверить. Точно так же можно провести небольшой эксперимент по проверке различных гипотез относительно причин неполадки. Предположим, требуется выяснить, с чем связана неполадка: с дефектом памяти или с перегревом процессора. Два возможных способа проверки таковы: заменить микросхемы памяти и установить более мощный вентилятор процессора. Эти проверки соответствуют двум потенциальным решениям проблемы, хотя не всегда получается именно так. Например, если память представлена двумя модулями DIMM (Dual In-line Memory Module — модуль памяти с двухрядным расположением выводов), попробуйте временно удалить сначала один, потом другой. Если один из них дефектный, источник проблемы будет найден.

Эмпирическое правило проведения экспериментов гласит, что за раз нужно менять только одну переменную. В частности, в рассмотренном выше случае не следует менять одновременно память и вентилятор, иначе не удастся точно установить источник проблемы. Но от правила можно отступать, если "подозреваемых" компонентов, допустим, четыре. Одновременная замена двух из них позволит сузить круг поиска и провести меньшее число экспериментов.

Важно также тщательно фиксировать предпринимаемые действия. Записывайте, какие проверки выполняются и каковы их результаты. Если не вести таких записей, то в ходе длительной диагностики можно легко забыть, была ли проведена та или иная проверка, что увеличит время, затрачиваемое на решение проблемы. Часто случается, что проблема, которая поначалу казалась простой, перерастает в настоящее "стихийное бедствие", поэтому не будьте беспечны. Как только возникнет настораживающая тенденция, составьте подробное описание неполадки, пока важные подробности не стерлись из памяти.

При диагностике программной проблемы попытайтесь по возможности воспроизвести ее в нескольких учетных записях. Подобные проблемы часто связаны с особыми конфигурационными настройками отдельных пользователей, поэтому важно находить такие "особенности".

Чрезвычайно полезным средством выявления неполадок являются журнальные файлы. Они обычно находятся в каталоге `/var/log` и хранят сообщения о действиях различных программ и подсистем. Наиболее важный из них — `/var/log/messages`. Остальные файлы связаны с `sendmail`, Samba, `cron` и другими программами. Как только проявились признаки неполадки, просмотрите окончание соответствующего журнального файла. Для этого существует команда `tail`. Она отображает несколько последних строк файла, которые в случае журнального файла являются самыми поздними по времени.

Аппаратная диагностика

Аппаратные проблемы сложно отслеживать, поскольку они имеют спорадический характер. Компьютер может давать случайный сбой, допустим, дважды в неделю, что существенно затруднит отладку, ведь вносимые изменения дадут результат лишь через несколько дней. Важно знать общие симптомы аппаратных неполадок, чтобы быстро ориентироваться в возникающих проблемах.

Для любых устройств важно проверять физический контакт с другими аппаратными компонентами. Процессор, модули памяти и платы расширения вставляются в гнезда на материнской плате, а кабели внешних устройств подключаются к разъемам материнской платы или плат расширения. Если эти соединения ослаблены, аппаратура может работать ненадежно или вообще не работать.

Проблемы, связанные с процессором

Хорошо, если дефект процессора проявится в невозможности загрузить компьютер. Такую проблему трудно не заметить, и ее решением является замена процессора.

Если компьютер не загружается, попробуйте удалить все внешние устройства, в частности звуковые и сетевые платы, дисководы CD-ROM и т.д. Не добившись результата, замените ключевые компоненты, такие как модули памяти, процессор и контроллер диска (если это отдельная плата). После замены дефектного компонента система должна загрузиться. Если ничего не получилось, значит, причину следует искать в материнской плате.

Более распространенная проблема связана с перегревом процессора. Все современные процессоры семейства x86 генерируют столько тепла, что могут уничтожить сами себя за считанные секунды. Чтобы избежать этого, производители компьютеров размещают поверх процессоров радиаторы и вентиляторные блоки (рис. 32.1). Радиатор сам по себе рассеивает достаточно тепла, пропуская воздух через охлаждающие ребра, а вентилятор усиливает прогонку воздуха через радиатор. Иногда крепление радиатора ослабевает, и его нужно усиливать с помощью двух защелок, расположенных по бокам радиатора. Случается также, что вентилятор перестает работать или скорость его вращения снижается из-за наслоений пыли. Скопление пыли вообще снижает эффективность охлаждения процессора. Во всех этих случаях компьютер может испытывать непреднамеренные сбои. Аналогичные симптомы появляются, когда температура в помещении становится слишком высокой (пороговое значение зависит от компьютера, но как правило, если пользователям жарко в помещении, то и для компьютера это вредно). Иногда перегрев можно ослабить, сняв крышку корпуса компьютера, но результат бывает и противоположным. Долгосрочным решением проблемы является за

мена радиатора и вентиляторного блока. Если в компьютере инсталлировано много жестких дисков и плат расширения, установите дополнительный вентилятор для охлаждения самого корпуса. Рекомендуемые модели вентиляторов можно узнать на Web-узле производителя процессоров. В случае, если радиатор и вентилятор длительное время работали недостаточно интенсивно, придется менять процессор, поскольку избыточное тепло наверилка вывело его из строя.

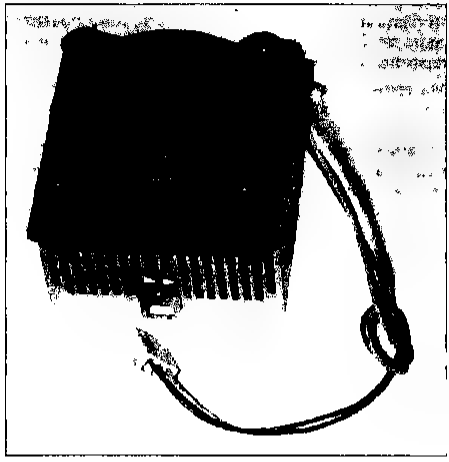


Рис. 32.1. Радиатор и вентиляторный блок способствуют отводу тепла, генерируемого процессором

Еще одна, к счастью самая редкая, проблема связана с тем, что современные процессоры слишком сложны и неизбежно содержат конструктивные недостатки. Обычно эти недочеты незначительны или хотя бы четко известны, что позволяет современным компиляторам легко обходить их. Но иногда производители допускают серьезные "проколы", как в случае с пресловутой ошибкой операций с плавающей запятой в процессорах Pentium середины 90-х годов или ошибками, которые вызывали сбой компиляторов GCC на ранних моделях процессоров AMD K6. Известия о таких ошибках молниеносно распространяются в Internet. Решением проблемы является замена процессора или обновление компилятора.

Проблемы, связанные с материнской платой

Материнские платы составляют "сердце" компьютера, поэтому их неполадки самым серьезным образом отражаются на надежности системы. Существует три класса проблем, связанных с материнскими платами:

- **Слишком новая модель платы.** Некоторые модели материнских плат вызывают проблемы во FreeBSD просто потому, что они слишком новые. В них используются наборы микросхем, для которых еще нет драйверов FreeBSD. Признаком таких проблем является невозможность использовать встроенные интерфейсы платы, например дисковый интерфейс EIDE, звуковой или видеоин-

терфейс. Иногда эти компоненты доступны, но демонстрируют неоптимальную производительность. Если доступна более новая версия FreeBSD, попробуйте установить ее. Если — нет, то проблемные компоненты всегда можно заместить платами расширения. Например, можно отключить EIDE-контроллер материнской платы и установить контроллер PCI EIDE.

- **Конструктивные недостатки** Подобно процессорам, материнские платы часто содержат конструктивные недостатки. Их тоже можно обойти за счет драйверов, но если плата новая, придется подождать, пока появится соответствующий драйвер. Такие проблемы часто проявляются в нестабильном доступе к встроенным интерфейсам материнской платы.
- **Поломки** Материнские платы могут повреждаться из-за статического электричества, сбоев питания или в результате неосторожного обращения. В большинстве таких случаев компьютер просто не загрузится, и этот симптом почти невозможно отличить от случая с поломкой процессора. Иногда повреждение отражается только на одной подсистеме, например на дисковом контроллере, тогда выходом из ситуации будет добавление замещающего компонента. Такие проблемы проявляются на компьютерах, которые до того работали нормально.

Как правило, неполадки материнской платы приводят либо к невозможности загрузить компьютер, либо к проблемам в работе некоторых подсистем, например дисковой. Замена материнской платы — довольно неприятная процедура, поэтому к ней прибегают в крайнем случае.

Проблемы, связанные с памятью

Дефектные модули памяти вызывают массу проблем. К их симптомам относятся внезапные сбои программ и непроизвольная перезагрузка системы. В редких случаях происходит повреждение документа, находящегося в памяти. Серьезно поврежденные модули выявляются материнской платой в ходе процедуры самотестирования при включении питания. Если компьютер "зависает" в процессе тестирования, память наверняка повреждена.

В программе настройки BIOS может быть включен режим ускоренного самотестирования. Если есть подозрение, что память работает неправильно, отключите ускоренную проверку и попробуйте загрузить компьютер, чтобы заставить его выполнить полноценную проверку.

Во FreeBSD есть программа `memtest`, выполняющая различные проверки памяти. Подобные проверки, естественно, нельзя считать исчерпывающими, но если уж программа сообщает о наличии проблемы, то это серьезно. Продавцы компьютерной техники располагают специальным оборудованием для тестирования модулей памяти, так что в случае необходимости можно прибегнуть к их услугам.

Модули памяти и другие компьютерные компоненты должны храниться в антистатических пакетах. Пластик, из которого делают такие пакеты, обладает антистатическими свойствами, что позволяет свести к минимуму риск повреждения компонента электростатическим зарядом.

Проблемы, связанные с дисками

Существует два источника проблем, связанных с жесткими дисками и другими типами дисковых устройств (например, CD-ROM и Zip).

- **Дисковый контроллер** Контроллер EIDE или SCSI-адаптер могут быть неисправными. Такие проблемы обычно отражаются на всех устройствах, управ-

ляемых контроллером, хотя бывают исключения. Иногда проблему удается преодолеть путем обновления ядра FreeBSD, если речь идет о конструктивном недостатке устройства или ошибочном драйвере

- **Диск**. Жесткие диски тоже могут выходить из строя. Такую проблему иногда можно решить за счет низкоуровневого форматирования диска, но чаще его просто меняют

Проблемы с дисками проявляются в снижении производительности дисковой подсистемы, неожиданных сбоях операций чтения/записи, повреждении файлов, невозможности запустить некоторые программы и даже в невозможности загрузить систему (сбой скорее всего произойдет на позднем этапе, когда загружено ядро). Если поврежденный сектор существует в разделе подкачки, начнут возникать сбои при обращении к памяти, как будто проблема связана с модулями памяти

Многие производители дисковых устройств предлагают программы тестирования, выполняющие полномасштабную проверку диска и сообщающие о ее результатах. К сожалению, эти программы в основном работают в DOS или Windows. DOS утилиты можно запускать из оболочки FreeDOS (`www.freedos.org`). В крайнем случае временно перенесите системный диск FreeBSD на Windows-компьютер и выполните проверку там

Некоторые утилиты тестирования дисков предлагают выполнять разрушающие и неразрушающие проверки. В первом случае происходит перезапись данных на диске, поэтому не применяйте такую проверку к диску, содержащему важные данные, включая инсталляцию FreeBSD.

Проблемы, связанные с видеоподсистемой

Наиболее распространенные проблемы с видеоподсистемой касаются базовой конфигурации пакета XFree86. Они могут проявляться в том, что X-среда не загружается или работает с минимальными разрешением и частотой обновления экрана. Конфигурирование X Window System рассматривалось в главе 13, поэтому прочитайте ее, чтобы узнать, как правильно настроить X-среду. Многие проблемы возникают при попытке использовать графическую плату с набором микросхем, который не поддерживается в текущей версии пакета XFree86. Обратитесь на Web-узел `www.xfree86.org`, чтобы узнать, какая версия пакета является наиболее новой и какая версия нужна для поддержки имеющейся графической платы. В случае необходимости обновите инсталляцию пакета. Более новый драйвер может быть также доступен на Web-узле производителя платы или используемого в ней микропроцессорного набора

Если внезапно перестает работать X Window System или пропадает видеосигнал, снимите крышку корпуса компьютера, чтобы проверить работу графической платы. Современные платы часто оснащены столь мощными микропроцессорами, что им, как и центральным процессорам, тоже нужны радиаторы, а иногда даже вентиляторы (см. рис. 32.1). Если вентилятор перестал вращаться, выключите компьютер и не включайте его до тех пор, пока неполадка не будет устранена.

Неполадки бывают не только у графических плат, но и у мониторов. Пользователи не должны заниматься их ремонтом, это задача специалистов. К счастью, проблемы мониторов не имеют отношения к FreeBSD

Мониторы содержат компоненты, которые способны длительное время сохранять высокое напряжение. Не пытайтесь починить монитор самостоятельно, так как можно получить сильный удар током. Лучше отнесите его в мастерскую

Одна из проблем которая может быть легко устранена пользователями связана с влиянием электромагнитных полей Люминесцентные лампы, неэкранированные звуковые колонки (большинство компьютерных аудиоколонок экранировано), телефоны и ряд других устройств создают магнитные поля, искажающие изображение на экране ЭЛТ-монитора или вызывающие его дрожание В этом случае достаточно переместить мешающее устройство или компьютер в другое место Жидкокристаллические мониторы не подвержены влиянию электромагнитных полей поэтому на крайний случай их можно использовать в качестве замены ЭЛТ-мониторам

Проблемы, связанные со звуковой платой

Звуковые платы не относятся к числу обязательных компонентов системы, поэтому их неполадки не требуют немедленного устранения. В основном эти неполадки сводятся к невозможности использовать звуковые утилиты. В большинстве случаев проблемы связаны с отсутствием необходимых драйверов. Воспользуйтесь утилитой `kldload`, чтобы загрузить драйвер, или перекомпилируйте ядро включив поддержку имеющегося звукового оборудования

Во FreeBSD одна из проблем заключается в том, что большинство звуковых плат работает в однозадачном режиме, т е если программа А обращается к звуковой плате, то программа Б которой нужно сделать то же самое, вынуждена будет перейти в режим ожидания Таким образом, возможен "захват" звуковой платы некорректно написанной или даже "зависшей" программой. Если звук какое то время был, а затем исчез и появился только после перезагрузки системы, наверняка имел место именно такой "захват". Предпринимаются попытки разработать интерфейсы для решения этой проблемы, но пока они далеки от завершения

Другая проблема проявляется в искажении звука, когда вместо ожидаемого звукового потока слышен неприятный резкий гул Иногда ее можно решить путем обновления звуковых драйверов ядра Эта проблема часто возникает при попытке воспроизвести звуковой файл иного типа, чем тот, что ожидается программой. Попробуйте настроить параметры утилиты `play` или другого звукового проигрывателя.

Проблемы, связанные с внешними устройствами ввода-вывода

К внешним устройствам ввода-вывода относится широкий круг устройств включая параллельные порты последовательные порты RS 232 и порты USB. Во FreeBSD для них существуют вполне стабильные драйверы (к USB это относится в меньшей степени) которые редко вызывают проблемы Наиболее вероятные источники неполадок таковы

- **Неподлежащие кабели.** В ряде случаев серьезные проблемы возникают из-за кабелей Для некоторых принтеров необходимы двунаправленные параллельные кабели, поэтому убедитесь в том, что выбран правильный тип принтерных кабелей Проблемы с последовательным портом RS-232 могут быть вызваны использованием нуль-модемного, а не прямого кабеля или наоборот поскольку они не всегда должным образом маркируются Кабели бывают просто изношены или повреждены
- **Дефектные разъемы** Разъемные соединения внешних интерфейсов могут повреждаться и выходить из строя, особенно если к порту часто подключаются различные устройства Самое простое решение в этой ситуации — добавить к компьютеру замещающий порт. Если поврежденный разъем относится к внешнему устройству, то нужно менять или чинить само устройство
- **Некорректные сетевые параметры** Устройства с интерфейсом RS-232 позволяют настраивать такие сетевые параметры как скорость передачи данных Убедитесь в том, что эти параметры заданны корректно в любой сетевой программе, в частности в терминальных программах и в обработчике очереди на печать

- **USB-драйверы определенных устройств**. Во FreeBSD есть универсальный USB драйвер, но драйверы нужны и конкретным USB-устройствам. Во FreeBSD 5.0 поддерживается не очень много таких устройств. Информация о драйверах для них доступна по адресу `http://www.etla.net/~n_hibma/usb/`.

Проблемы, связанные с сетевыми устройствами

Неполадки сетевого оборудования сложно отслеживать, поскольку их причины могут быть самыми разными. Рассмотрим некоторые нюансы, на которые следует обращать внимание.

- **Физические соединения**. Сетевые кабели могут быть повреждены или дефектны как и любые другие внешние кабели. Во многих сетях компьютеры соединяются посредством коммутаторов и концентраторов, которые тоже подвержены поломкам. Замена кабеля — простая, но важная и, главное, недорогая проверка, которую следует выполнять, когда сетевое соединение полностью перестает функционировать.
- **Индикаторные лампочки**. Сетевые платы, концентраторы и коммутаторы имеют индикаторные лампочки, загорающиеся в момент установления соединения, а также в процессе передачи графика. Это великолепное средство диагностики. Лампочки позволяют выявлять проблемы с физическими соединениями (в этом случае индикаторные лампочки будут выключены) и контролировать факт передачи данных. Если данные поступают в сеть, то проблема может быть связана как с аппаратным, так и с программным обеспечением.
- **Совместимость драйверов**. Важную роль играет совместимость драйверов различных устройств. Это особенно важно для некоторых видов Ethernet-плат, в частности тех, что основаны на микропроцессорном наборе DEC/Intel "Tulip", и их клонов. Суть проблемы заключается в последней фразе "и их клонов". Существует слишком много разновидностей микропроцессорного набора Tulip, поэтому разработчики FreeBSD испытывают определенные трудности с ними. Сетевые платы Ethernet достаточно дешевы, так что проще заменить плату, чем пытаться заставить работать ее драйвер.

Многие сетевые проблемы имеют программную природу: некорректные параметры, переданные команде `ifconfig`, несовместимость клиентов и серверов и т.д. Обо всем этом будет подробнее говориться на с. 641. Одна из возможных проверок — просто ввести **`ifconfig`** или дополнительно указать имя интерфейса. Команда выдает информацию о нужном интерфейсе:

```
# ifconfig vr0
vr0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet 192.168.1.6 netmask 0xffffff00 broadcast 192.168.1.255
        ether 00:80:c8:fa:3b:0a
        media: Ethernet autoselect (100baseTX)
        status: active
```

Как видно из полученного отчета, система обнаружила сетевую плату и вполне разумно ее сконфигурировала. Если подобный отчет не выдается, то плата либо неправильно вставлена в гнездо, либо полностью неисправна, либо не распознана ядром (возможно, это новая модель, для которой еще нет драйверов FreeBSD). Если же плата распознается, но ей не назначаются IP-адрес и сетевая маска, проблема может заключаться в соединении с DHCP-сервером.

Проблемы, связанные с принтером

Проблемы, касающиеся принтеров, часто бывают вызваны несовместимостью с программой Ghostscript. Как описывалось в главе 9, FreeBSD использует эту програм-

му для преобразования PostScript-данных генерируемых программами печати в формат понятный имеющемуся принтеру. Чтобы узнать какие принтеры работают с программой Ghostscript, обратитесь на Web-узел `www.linuxprinting.org`

Некоторые проблемы могут быть связаны с внешними интерфейсами или кабелями как указывалось на с. 638. Для выявления таких проблем проверьте функционирование индикаторных лампочек принтера, которые должны мигать при передаче данных принтеру, даже если задание не печатается корректно

Программная диагностика

Диагностика программных проблем часто заключается в проверке конфигурационных файлов или сценариев запуска системы Многие неполадки являются следствием ошибок в программах поэтому их устранение сводится к обновлению программ или библиотек

Проблемы, связанные с запуском системы

В стандартном варианте инсталляции FreeBSD должна загружаться корректно хотя существует вероятность того, что из-за неполадок отдельные компоненты не будут работать. Например, если сетевая конфигурация системы неправильна, сетевые утилиты откажутся функционировать Проблемы с запуском системы могут вызываться тремя различными причинами

- **Системный загрузчик** Загрузчик системы рассматривавшийся в главах 3 и 4 позволяет выбрать операционную систему для запуска. Даже если на компьютере установлена только FreeBSD, загрузчик необходим, просто в этом случае он непосредственно запускает FreeBSD Проблемы с загрузчиком проявляются после завершения процедуры самотестирования BIOS, но перед началом работы ядра, когда компьютер "зависает", отобразив лишь пустой экран или одиночное сообщение об ошибке. Это может произойти сразу после инсталляции FreeBSD или изменения конфигурации загрузчика. Сходные проблемы могут возникнуть и после инсталляции другой операционной системы (особенно этим славятся" системы компании Microsoft)
- **Ядро**. Если ядро FreeBSD содержит ошибки, неправильно сконфигурировано или не имеет необходимых драйверов, это может повлиять на загрузку компьютера или работу отдельных аппаратных устройств. Сбой загрузки наиболее вероятен, когда ядро проверяет список имеющихся устройств В случае стандартного ядра FreeBSD нужно либо обновить ядро либо заменить проблемное устройство Если же проблема возникает при подключении пользовательского ядра, восстановите стандартное ядро, пока неполадка не будет устранена. Скорее всего, в новое ядро не включен обязательный драйвер. Подробнее о конфигурировании ядра рассказывалось в главе 12
- **Сценарии запуска** Сценарии запуска системы бывают неполными или ошибочными Стандартные сценарии являются хорошей отправной точкой однако в них могут задаваться неверные опции Ошибки сценариев проявляются после того, как ядро завершило проверку устройств. С этого момента FreeBSD начинает отображать сообщения о запуске конкретных программ, например `sendmail` или `cron`. Некоторые неполадки остаются незамеченными вплоть до окончания загрузки системы, когда оказывается, что какая-нибудь подсистема работает совсем не так, как ожидалось. Если тот или иной сетевой сервер не запускается, просмотрите инструкции сценариев, касающиеся его запуска, и убедитесь в отсутствии опечаток На с. 641 будет приведена дополнительная информация о причинах сетевых неполадок

Ошибки запуска неприятны так как они либо вообще препятствуют загрузке системы, либо приводят к тому что загруженная система работает неоптимальным образом. Во многих случаях решить проблему можно за счет аварийной системы, которая обеспечит минимально необходимое функционирование FreeBSD Другое решение — инсталлировать загрузчик другой операционной системы вместо поврежденного или неправильно работающего FreeBSD-загрузчика этапа 0.

Проблемы, связанные с драйверами устройств

Иногда загрузке системы препятствуют неполадки драйверов устройств. Например если в системе установлен единственный жесткий диск SCSI, но нет драйверов для SCSI-адаптера, FreeBSD не загрузится Возможно, появится несколько сообщений о загрузке ядра, но до сценариев запуска системы дело не дойдет Такие проблемы возникают в процессе инсталляции системы, а также при подключении только что скомпилированного ядра.

С драйверами связан целый ряд менее серьезных проблем, хотя выражение менее серьезный" в данном контексте означает довольно неприятные события, например непроизвольные сбои системы К числу других проблем относятся: недоступность устройств, не влияющих на загрузку системы например звуковой или сетевой платы ошибки при работе с устройствами, такие как искажение звука; плохая производительность, в частности медленный доступ к диску. Короче говоря, неполадки драйверов проявляются примерно так же, как и неисправности в работе устройств.

Диагностика таких неполадок в основном сводится к определению их природы т е к поиску ответа на вопрос, на каком уровне – аппаратном или программном — существует проблема. Для этого можно переставлять устройства на другие компьютеры и проверять их в различных операционных системах, но учтите, что подобные проверки иногда вводят в заблуждение Например, две операционные системы могут содержать одинаковые ошибки в драйверах устройств особенно если это родственные системы

Чтобы устранить неполадки с драйверами, нужно либо обновить ядро до той версии, которая содержит исправленные драйверы, либо перекомпилировать ядро, включив в него отсутствующий драйвер. В некоторых случаях достаточно всего лишь загрузить модуль драйвера с помощью команды `kldload` (см главу 12).

Проблемы, связанные с сетевым обеспечением

Сетевые проблемы трудно отслеживать, поскольку сетевая подсистема довольно сложна. Некоторые неполадки могут вызываться сетевым оборудованием или драйверами, но в основном они связаны с конфигурацией программного обеспечения. Рассмотрим главные источники неполадок

- **Сетевые устройства и драйверы** Сетевые устройства могут быть неисправными а драйверы — содержать ошибки. Если сетевое соединение постоянно сбоит причина может быть именно в этом
- **Сценарии запуска сетевой подсистемы**. Сценарий `/etc/network` управляет запуском сетевых процессов Его работа зависит от установок других сценариев главный из которых — `/etc/rc.conf`. Именно в этих файлах определяется основная конфигурация сетевых интерфейсов, а также содержатся команды запуска многих стандартных серверов. Опечатки и другие конфигурационные ошибки, присутствующие в этих файлах, могут вызывать целый ряд проблем — от недоступности сети до непоиятных сбоев в работе серверов
- **Серверные конфигурационные файлы**. У большинства серверов есть собственные конфигурационные файлы в которых задаются параметры их работы Если ка

кой-то сервер функционирует не так, как ожидается начните поиск причин с анализа конфигурационных и журнальных файлов

- **Брандмауэры** Если в системе сконфигурирован брандмауэр (см главу 17), он вмешивается в работу сетевых клиентов и серверов. Типичные признаки такой ситуации — когда клиенту не удается связаться с внешней системой или сервер не отвечает на запросы от внешних систем Проблемы могут также возникать из-за брандмауэра, работающего на другом компьютере. Например на маршрутизаторах иногда включают фильтрацию пакетов, чтобы заблокировать доступ к определенным портам из соображений безопасности
- **DNS.** Работа глобальной сети во многом зависит от DNS (Domain Name System — система доменных имен), которая преобразует имена компьютеров в IP-адреса и наоборот. Распространенная проблема конфигурации заключается в неправильном задании DNS сервера В файле `/etc/resolv.conf` определяется от одного до трех таких серверов Для каждого из них в строке `nameserver` указывается IP-адрес Если все эти адреса недействительны или соответствующие DNS-серверы не функционируют, нельзя будет получить дистанционный доступ к компьютерам по именам, хотя IP-адреса по прежнему будут восприниматься
- **Ошибки отдельных программ** В программном коде как клиентов, так и серверов могут быть ошибки, вызывающие разного рода сбои Стандартным решением проблемы является обновление программы

Ключом к сетевой диагностике является воспроизведение неполадки на нескольких компьютерах Например, если внешний Web-узел недоступен с любого компьютера локальной сети, то проблема заключается в самом внешнем компьютере или в правилах фильтрации пакетов, но не в настройках Web-броузера Если неполадка возникает лишь в некоторых системах, выясните, что у них общего. Возможно, во всех этих системах инсталлирована одна и та же версия броузера или все они находятся в одной подсети Поиск общих черт играет важную роль в выяснении причины неполадок

Проблемы, связанные с пользовательскими программами

Программы, запускаемые рядовыми пользователями, тоже могут содержать ошибки. Их диапазон чрезвычайно широк. В то же время совершенно очевидно, что сфера действия таких ошибок ограничена конкретной программой, которая и будет демонстрировать ненормальное поведение. Исключение составляют лишь библиотеки функций, совместно используемые многими программами.

Когда от пользователя поступает отчет о возникшей проблеме, в первую очередь попытайтесь воспроизвести ее в новой учетной записи Некоторые неполадки обусловлены ошибками в пользовательских конфигурационных файлах В таком случае тестирование программы в новой среде не выявит проблем, что сузит круг возможных причин неполадки до ошибок конфигурации Проверьте файлы конфигурации пользователя, а также права доступа к его файлам и каталогам, включая групповые права доступа Воспользуйтесь командой `diff`, чтобы сравнить его конфигурационные файлы с эквивалентными файлами новой учетной записи и выяснить суть отличий

Если те же самые проблемы возникают и в новой учетной записи, проверьте программу к которой они относятся. Возможно, ошибка присутствует в ее программном коде или в глобальном конфигурационном файле. В случае необходимости поищите обновленную версию программы.

Ряд проблем связан с правами доступа Например, программам работы со звуком нужен доступ к файлам устройств звуковой платы. В стандартной инсталляции FreeBSD 5.0 рядовым пользователям разрешена запись в эти файлы, но не исключено, что такая конфигурация была отменена администратором Кроме того, не все про-

граммы располагают аналогичными привилегиями В частности, утилитам работы с магнитными лентами и дисководами CD-R требуется доступ к соответствующим устройствам, тогда как по умолчанию он есть только у суперпользователя

Вполне возможно, что вы — первый, кто столкнулся с подобной ошибкой. Если она довольно специфична, — например, возникает только при определенных, весьма нетипичных обстоятельствах, — свяжитесь с автором программы. Более подробные указания обычно приводятся на официальном Web-узле программы Иногда отчет об ошибке нужно посылать по электронной почте Для крупных проектов предусмотрены Web-ориентированные системы составления отчетов об ошибках Если вы хорошо разбираетесь в языке программирования, а также инструментальных средствах, использованных для написания программы, попробуйте выполнить отладку самостоятельно и сообщить найденное решение автору Это прекрасный способ помочь сообществу разработчиков программ с открытым исходным кодом — сообществу, которое создало FreeBSD

Тестирование полученного результата

Во многих случаях обнаружение неполадки и ее устранение — взаимосвязанные операции Методы диагностики часто предполагают внесение изменений, которые, в случае успешной идентификации проблемы, заодно и решают ее Но после того как неполадка устранена, необходимо еще проверить модифицированную подсистему Например, если программа часто сбоила и пришлось устанавливать более стабильную ее версию, нужно исследовать новую программу, скажем, запустить ее на длительное время или "прогнать" через большой массив входных значений. Такие проверки позволят убедиться в том, что проблема действительно решена и не повторится вновь

Некоторые неполадки трудно диагностировать и устранять, поскольку они имеют спорадический характер После внесения изменений, которые, как предполагается решают проблему следует провести тщательное тестирование. В простейшем случае достаточно оставить компьютер работающим, в надежде, что неполадка не повторится. В других ситуациях требуется проверить журнальные файлы, загруженность центрального процессора или другие показатели производительности системы Чтобы упростить этот процесс, создайте задание демона `cron` или воспользуйтесь программой типа SWATCH, которая будет искать в журнальных файлах заданные образцы записей.

Если найденное решение не дает ожидаемого результата, вернитесь к тому, с чего начали Иногда исходное решение является правильным, просто недостаточно радикальным Например, если заменить вентилятор перегревающегося процессора моделью, которая лучше исходной, но все равно не отводит избыточное тепло полностью, сбои системы не прекратятся, а лишь станут менее частыми Решением проблемы будет установка еще более мощного радиатора или вентилятора Для неисправной программы может также существовать обновление, которое не устраняет ошибку и потому не дает нужного результата

Использование аварийной системы

Некоторые неполадки столь серьезны, что препятствуют загрузке системы или же требуют редактирования файлов таким способом, который невозможен при нормальной работе FreeBSD В этом случае на помощь приходит аварийная загрузочная система, которая позволяет получить доступ к FreeBSD, не загружая ее Аварийные системы полезны также для выполнения определенных видов обновлений и процедур об

служивания таких как изменение размера корневого раздела Существуют следующие разновидности аварийных систем.

- **Инсталляция другой операционной системы** Иногда для аварийного восстановления FreeBSD удобно применять другую операционную систему. К примеру, можно использовать дисковую утилиту любой операционной системы для проверки таблицы разделов *x86* (правда, лишь некоторые операционные системы, не относящиеся к семейству UNIX, распознают подразделы, созданные в первичном разделе FreeBSD). Если загрузчик этапа 0 поврежден, разрешается заменять его внешними загрузчиками, такими как LILO или коммерческая программа BootMagic компании PowerQuest (`www.powerquest.com`).

Внимание Ряд операционных систем, в частности Linux поддерживает файловые системы FreeBSD. Но несмотря на заявления о поддержке, следует с осторожностью подходить к записи данных в разделы FreeBSD. В той же Linux средства поддержки FFS/UFS еще весьма несовершенны и могут вызывать повреждение файловой системы.

- **Альтернативная инсталляция FreeBSD** Можно инсталлировать FreeBSD дважды на один компьютер. Одна инсталляция будет основной, а другая — служебной содержащей лишь минимальный набор утилит Вторую систему можно загружать в аварийных ситуациях и для выполнения низкоуровневого обслуживания
- **Инсталляция на другом жестком диске** Можно инсталлировать FreeBSD на другой жесткий диск и использовать ее для обслуживания основного жесткого диска Если второй жесткий диск присутствует постоянно, это будет эквивалентно предыдущему случаю просто такую инсталляцию не обязательно планировать с самого начала
- **Инсталляция на съемном носителе**. Другой вариант заключается в инсталляции FreeBSD на высокоемкий съемный диск, такой как Zip, LS-120 или Jaz. Подготовьте и протестируйте такую систему, а затем выньте диск и спрячьте его, чтобы воспользоваться в случае возникновения аварийной ситуации. Сложность этого подхода состоит в том, что указанные устройства могут не быть загрузочными поэтому придется особым образом конфигурировать системный загрузчик
- **Загрузочный компакт-диск FreeBSD** В той версии FreeBSD, которая распространяется на нескольких компакт-дисках, второй диск является загрузочным По умолчанию загружается нормальная инсталляция системы, но если выбрать опцию `Fixit`, будет запущена система, которая образована стандартным набором утилит, содержащихся на компакт-диске. Основную систему нужно смонтировать, чтобы получить возможность редактировать файлы, выполнять служебные утилиты и т.д.

В общем случае наиболее удобен последний вариант Загрузочный компакт диск FreeBSD содержит небольшой, но вполне работоспособный вариант системы, включая редакторы `vi` и `ee`, стандартные дисковые утилиты и команды, такие как `mount` и `fsck`, а также утилиты архивирования и резервного копирования например `tar` Смонтировав основную систему, можно запускать по крайней мере некоторые ее программы (программы, работающие с библиотеками функций, могут не находить их)

Совет Изучите возможности доступных аварийных систем Попробуйте загрузить такую систему, а затем прочитать и отредактировать несколько малозначительных файлов основной системы (или, для большей безопасности тестовой системы) Это нужно для того, чтобы в случае возникновения реальной аварийной ситуации не нанести системе еще больший вред, применяя утилиты восстановления

Резюме

Существуют различные эвристические правила и практические приемы, упрощающие поиск и устранение неполадок в системе Круг возможных проблем и их решений столь велик, что в рамках одной главы можно рассмотреть лишь важнейшие признаки наиболее распространенных проблем и ключевые подходы к их решению В особенно трудных ситуациях обращайтесь за помощью к дополнительным источникам информации таким как специализированные Web-узлы и группы новостей Usenet.

Словарь терминов

- **AGP (Accelerated Graphics Port)** – аппаратный стандарт подключения графических плат к материнской плате. У большинства современных материнских плат, выпущенных для платформы x86, есть только одна шина AGP, но иногда на самой плате устанавливают графический микропроцессор, имеющий внутреннюю шину AGP. Старые материнские платы использовали для этих целей шину *PCI* или *ISA*.
- **ASCII (American Standard Code for Information Interchange)** — американский стандартный код обмена информацией, определяющий числовое кодирование алфавитно-цифровых символов, знаков препинания и определенных управляющих символов.
- **ATA (Advanced Technology Attachment)** — аппаратный стандарт подключения жестких дисков и аналогичных устройств к платам контроллеров и материнским платам. Также известен как *EIDE*.
- **ATAPI (ATA Packet Interface)** — аппаратный интерфейс дисководов CD-ROM, ленточных дисководов и ряда других устройств, использующих стандарт *ATA*.
- **BSD (Berkeley Software Distribution)** — название исправлений к исходной версии AT&T UNIX, а впоследствии — полноценной операционной системы, от которой ведет начало FreeBSD.
- **BSD-лицензия** — лицензия на программное обеспечение, в соответствии с условиями которой распространяется большая часть программ FreeBSD. Разрешает распространение модифицированной версии программного продукта на условиях лицензии BSD или другой лицензии. См. также *GPL* и *Открытый исходный код*.
- **CHS (Cylinder/Head/Sector)** — схема адресации "цилиндр–головка–сектор" (ЦГС), определяемая числовыми триплетами, которые уникальным образом идентифицируют каждый сектор (обычно 512 байт) данных на жестком диске. Применяется в жестких дисках EIDE, а также в структурах данных, описывающих разделы x86. В современных жестких дисках номера цилиндров, головок и секторов являются удобными фикциями, не имеющими ничего общего с истинной геометрией жесткого диска. См. также *LBA*.
- **CIFS (Common Internet File System)** См. *SMB/CIFS*
- **CSS (Content Scrambling System)** — метод шифрования данных, применяемый в коммерческих DVD-дисках.
- **DHCP (Dynamic Host Resolution Protocol)** – протокол динамического конфигурирования компьютеров, позволяющий серверу автоматически предоставлять информацию, касающуюся протоколов TCP/IP, своим клиентам. Применение DHCP существенно упрощает сетевое конфигурирование клиентских компьютеров.
- **EIDE (Enhanced Integrated Drive Electronics)** См. *ATA*
- **Ethernet** — распространенный тип локального сетевого оборудования со скоростью передачи данных от 10 Мбит/с до 1 Гбит/с, в зависимости от разновидности.
- **Ghostscript** — программа, преобразующая данные формата PostScript в другие графические форматы, включая форматы, поддерживаемые многими распространенными принтерами. Обычно вызывается программами-фильтрами в процессе обработки *очереди на печать* во FreeBSD.

- **GID (Group ID)** — число идентифицирующее группу пользователей Ему со поставлено имя группы
- **GNOME (GNU Network Object Model Environment)** — популярная графическая оболочка для FreeBSD. См также *KDE*.
- **GPL (General Public License)** лицензия на программное обеспечение, в соот ветствии с условиями которой распространяется часть программ FreeBSD Требу ет, чтобы модифицированные версии программного продукта тоже распростра нялись на условиях GPL. См также *BSD-лицензия* и *Открытый исходный код*
- **HTML (Hypertext Markup Language)** — способ форматирования текстовых фай лов Чаще всего применяется для составления Web-страниц
- **HTTP (Hypertext Transfer Protocol)** — протокол используемый Web броузерами и Web-серверами для доставки Web-страниц
- **ICMP (Internet Control Message Protocol)** — протокол передачи очень простых управляющих пакетов между компьютерами в сетях TCP/IP
- **Internet** — глобальная сеть, которая начала сверхбыстрыми темпами развивать ся в 90-х годах прошлого века. Позволяет пользователям всего мира обмени ваться электронной почтой, читать Web-страницы и многое другое
- **IP-адрес** — 4-байтовый (32-битовый) адрес компьютера в сети TCP/IP Обычно записывается в виде четырех десятичных чисел, разделенных точками, напри мер 172 19 65 11.
- **IRC (Internet Relay Chat)** – протокол общения пользователей в реальном вре мени.
- **ISA (Industry Standard Architecture)** — аппаратный стандарт подключения плат расширения, например звуковых, к материнской плате В настоящее время шина ISA почти не применяется, будучи вытесненной шиной *PCI*
- **KDE (K Desktop Environment)** популярная графическая оболочка для FreeBSD. См также GNOME
- **Kerberos** – протокол сетевой аутентификации и безопасности Позволяет вве сти пароль один раз в ходе сеанса, после чего пользоваться услугами любого Kerberos-совместимого сервера в сети. Дополнительную информацию можно найти по адресу `http://web.mit.edu/kerberos/www/`
- **LBA (Logical Block Addressing)** — линейная схема адресации жесткого диска, в которой каждый сектор (обычно 512 байт) данных уникальным образом иден тифицируется единственным числом. Всегда применяется в жестких дисках SCSI, поддерживается и современными дисками EIDE Альтернативная схема адресации — *CHS* — используется в старых дисках, а также в некоторых низ коуровневых структурах дисковых данных
- **MBR (Master Boot Record)** первый сектор жесткого диска на платформе x86 называемый главной загрузочной записью (*ГЗЗ*) Содержит код системного загрузчика и основную таблицу разделов
- **MIME (Multipurpose Internet Mail Extension)** — стандарт кодирования, исполь зуемый для идентификации типов файлов Коды MIME проверяются диспет черами файлов, а также при получении почтовых вложений и загрузке Web страниц
- **NIC (Network Interface Card)** — сетевая плата, вставляемая в компьютер и реа лизующая сетевой интерфейс (как правило, *Ethernet*)
- **PAM (Pluggable Authentication Module)** подключаемый модуль аутентифика ции применяемый для обработки имен пользователей и паролей во FreeBSD

Вместо того чтобы непосредственно читать файл `/etc/passwd` для проверки достоверности пользовательского пароля, программа вызывает нужный аутентификационный модуль, который выполняет эти действия стандартизированным способом

- **PCI (Peripheral Component Interconnect)** — аппаратный стандарт подключения плат расширения, таких как SCSI-адаптеры и сетевые платы, к материнской плате. В настоящее время широко распространен. См. также *ISA* и *AGP*.
- **PID (Process ID)** — число, уникальным образом идентифицирующее процесс. С идентификаторами процессов работают некоторые административные команды
- **PostScript** — язык описания страниц, разработанный компанией Adobe. Применяется в принтерах высокого класса (особенно в лазерных принтерах и фотонаборных машинах) и фактически является стандартным языком команд печати для программ UNIX (а значит, и FreeBSD). См. также *Ghostscript*.
- `root` — имя пользователя, соответствующее системному администратору (на практике администратор должен также иметь обычную учетную запись).
- **Samba** — сервер файлов и печати, работающий по протоколам *SMB/CIFS*, которые используются для организации совместного доступа к файлам и принтерам в среде Windows. Позволяет FreeBSD-системе играть роль файлового сервера или сервера печати для компьютеров Windows.
- **SCSI (Small Computer System Interface)** — аппаратный стандарт подключения жестких дисков и аналогичных устройств к компьютеру. Менее популярный, более дорогой и более производительный, чем конкурирующий стандарт *ATA*
- **SMB/CIFS (Server Message Block / Common Internet File System)** — протоколы организации совместного доступа к файлам и принтерам в среде Windows. См. также *Samba*
- **sticky-бит** — специальный бит прав доступа, который может устанавливаться для совместно используемых каталогов, таких как `/tmp`. Указывает на то, что только владелец файла имеет право удалить файл из такого каталога
- **TCP (Transmission Control Protocol)** — сетевой протокол, являющийся одним из важнейших компонентов стека *TCP/IP*. Пакеты TCP инкапсулируют данные, передаваемые в ходе соединения. Такие пакеты используются большинством распространенных протоколов Internet, включая SMTP, *HTTP*, Telnet и FTP
- **TCP/IP (Transmission Control Protocol / Internet Protocol)** — стек сетевых протоколов, наиболее широко используемый в настоящее время. Компьютеры, в которых реализован этот стек, могут взаимодействовать друг с другом по сети при наличии совместимых высокоуровневых программ (таких как сервер и клиент Telnet)
- **UDP (User Datagram Protocol)** — протокол обмена данными в сетях *TCP/IP*. Пакеты UDP более сложны, чем пакеты *ICMP*, но менее сложны, чем пакеты *TCP*. Они используются в ряде высокоуровневых протоколов, таких как DNS, *DHCP* и NFS
- **UID (User ID)** — число, идентифицирующее конкретную учетную запись. Ему сопоставлено имя пользователя.
- **USB (Universal Serial Bus)** — аппаратный стандарт подключения модемов, мышей, клавиатур, принтеров, сканеров и других низкоскоростных внешних устройств к компьютерам. Средства поддержки во FreeBSD пока еще ограничены

- **VNC (Virtual Network Computing)** — клиент серверный пакет, реализующий дистанционный доступ в графическом режиме к компьютеру, на котором есть XFree86, Windows, Mac OS или другая графическая среда
- **X Window System** — графическая среда FreeBSD, часто называемая X средой
- **Администратор системный** - человек, отвечающий за конфигурирование компьютера, в том числе добавление и удаление программ, создание пользовательских учетных записей и т.д. Также известен как *суперпользователь*. Использует учетную запись `root`.
- **Администрирование системное** действия, связанные с инсталляцией или обслуживанием операционной системы в целом, а также оборудования, на котором она работает. К типичным административным задачам относятся: удаление, обновление и удаление программ, создание и удаление пользовательских учетных записей; настройка параметров серверов.
- **Анализатор пакетов** программа, отслеживающая и при необходимости регистрирующая низкоуровневый сетевой график (пакеты данных). Удобное средство сетевой диагностики, а также инструмент взлома, применяемый для кражи паролей и других конфиденциальных данных
- **Ассемблер** программа преобразующая данные на языке ассемблера в машинный код
- **Ассемблера язык** — представление *машинного языка* в мнемонических кодах для удобства программирования. Программисты могут писать программы на языке ассемблера и преобразовывать их в машинные коды с помощью программы транслятора: *ассемблера*. Язык ассемблера имеет более низкий уровень, чем другие популярные языки программирования, такие как C, Pascal или Perl.
- **Бинарный** (1) См. *Двоичное представление*. (2) Формат файла, хранящего нетекстовые данные, например изображение или звук. (3) Формат исполняемого файла программы. *Исходный код* превращается в бинарную форму путем компиляции
- **Бинарный код**. См. *Машинный язык*. (Термин "бинарный код" является синонимом, поскольку процессор работает с двоичными числами, хотя на практике машинные коды часто представляются шестнадцатеричными числами для удобства восприятия.)
- **Бит** — двоичная цифра (т.е. 0 или 1) в двоичной системе счисления
- **Битовый образ**. См. *Растр*.
- **Брандмауэр** — сетевое устройство, компьютер или программа, задача которого/которой состоит в том, чтобы ограничивать входящий или исходящий сетевой трафик на основании IP адресов, типов пакетов либо других критериев. См. также *Фильтр пакетов* и *Прокси-сервер*
- **Векторная графика** — метод описания изображения в виде линий, кривых и других графических примитивов. Векторные изображения хорошо масштабируются на устройствах вывода при любом разрешении, в отличие от *растров*.
- **Взломщик** — человек, получающий несанкционированный доступ к компьютеру или намеренно нарушающий его нормальное функционирование. Мотивы для таких действий могут быть разными: любопытство, жадность, месть и т.д. В средствах массовой информации взломщиков обычно называют *хакерами*, хотя этот термин имеет более широкое значение
- **ГЗЗ (главная загрузочная запись)**. См. *MBR*.

- **Группа** — совокупность учетных записей. Каждая учетная запись принадлежит минимум одной группе. Группы могут применяться для предоставления нужным пользователям прав доступа к определенным ресурсам.
- **Двоичное представление** — система счисления по основанию 2, в которой каждая цифра равна либо 0, либо 1. Компьютеры оперируют именно двоичными числами.
- **Демон** — программа, работающая в фоновом режиме для предоставления определенных услуг. Большинство серверных программ функционирует в виде демонов.
- **Доменное имя** — имя, идентифицирующее фрагмент сети Internet, который находится под контролем одной организации или одного человека. См. также *Имя узла* и *Машинное имя*.
- **Доменное имя, полностью определенное** — имя компьютера, дополненное доменным именем, например `www.whitehouse.gov`.
- **Дочерний** — в контексте процессов FreeBSD означает процесс, запускаемый другим процессом. У программы может быть несколько дочерних процессов. См. также *Родительский*.
- **Загруженность средняя** — мера использования центрального процессора. Средняя загруженность 1,0 означает полную занятость процессора. Более высокие значения говорят о том, что число запросов от программ превышает возможности процессора, поэтому производительность большинства программ будет не оптимальной.
- **Загрузчик** — программа, управляющая процессом загрузки системы. Основной загрузчик позволяет пользователю выбрать операционную систему, после чего передает управление вторичному загрузчику.
- **Загрузчик этапа *n*** — Во FreeBSD используется трехэтапная процедура начальной загрузки. Загрузчик первого этапа находится в главной загрузочной записи (ГЗЗ) жесткого диска и позволяет выбрать операционную систему. Загрузчики второго и третьего этапов находятся в первичном разделе FreeBSD и позволяют задавать параметры ядра, а также загружать FreeBSD.
- **Зомби** (1) Компьютер, незаконным путем вовлеченный в распределенную атаку типа "*отказ от обслуживания*". (2) Процесс, который не был корректно завершен.
- **Имя домена**. См. *Доменное имя*.
- **Имя машинное**. См. *Машинное имя*.
- **Имя узла** — алфавитно-цифровое имя, назначенное компьютеру. В TCP/IP состоит из двух частей *машинного имени* и *доменного имени*.
- **Индексный дескриптор** — дисковая структура данных, используемая в FFS и некоторых других файловых системах. Индексные дескрипторы назначаются файлам, по одному на каждый файл. Элемент каталожного списка ссылается именно на индексный дескриптор, который содержит информацию о размере файла, месте его хранения на диске и т. д.
- **Интерпретатор** — программа, выполняющая файл *исходного кода* строка за строкой. Интерпретаторы выполняют программы относительно медленно, зато с интерпретируемыми языками проще работать, поскольку отсутствует этап компиляции. *Сценарии* — это программы, написанные на интерпретируемых языках.
- **Интерпретатор команд** — программа, принимающая команды от пользователя. Обычно это текстовая программа, в которой вводятся имена программных

файлов. Интерпретаторы команд FreeBSD поддерживают собственные языки сценариев, позволяющие автоматизировать выполнение рутинных действий

- **Исходный код** — форма компьютерной программы, создаваемая и модифицируемая программистом. Файлы исходных кодов либо непосредственно обрабатываются с помощью *интерпретатора*, либо преобразуются в *бинарный* формат для последующего автономного выполнения. См. также *Открытый исходный код*
- **Канал** — способ соединения программ в виртуальную цепочку, каждая программа в которой обрабатывает выходные данные предыдущей программы и передает полученные результаты следующей программе. См. также *Переадресация*
- **Клиент**. (1) Программа, посылающая запрос на передачу данных по сети. Примерами являются FTP-клиенты, программы чтения почты и Web-броузеры. (2) Компьютер, на котором работают сетевые клиентские программы, но отсутствуют серверные программы.
- **Коммутатор** — сетевое устройство, соединяющее несколько компьютеров. Отличается от *концентратора* тем, что передает исходные пакеты данных только получателю, а не всем подключенным устройствам. Это позволяет повысить производительность
- **Компилятор** — программа, преобразующая *исходный код* в *бинарную* форму, которая может выполняться компьютером
- **Консоль** — клавиатура, монитор и, возможно, мышь, подключенные к компьютеру. Чаще всего этим термином обозначают совокупность устройств ввода-вывода, подключенных *непосредственно* к компьютеру, в отличие от удаленных терминалов.
- **Консоль виртуальная** — текстовый сеанс, обслуживаемый независимо от других с одиночной консоли. Можно переключаться между виртуальными консолями FreeBSD, нажимая функциональные клавиши (от <F1> до <F8>, в X-среде нужно дополнительно нажимать клавишу <Alt>). Это позволяет запускать несколько программ переднего плана одновременно и независимо друг от друга
- **Концентратор** — сетевое устройство, соединяющее несколько компьютеров. Отличается от *коммутатора* тем, что передает пакеты данных всем подключенным устройствам, а не только адресату. Если два устройства одновременно посылают данные через концентратор, возникает "коллизия", при которой пакеты повреждают друг друга. В результате снижается производительность, поскольку отправителям приходится делать паузу, дожидаясь момента повторной отправки пакетов
- **ЛВС (локальная вычислительная сеть)** — сеть, охватывающая небольшую (территориально) область, например офис, здание или группу из нескольких зданий. Может подключаться к другим ЛВС, а также к Internet.
- **Маршрут стандартный** — спецификация, определяющая, куда следует посылать сетевые пакеты, для которых в таблице маршрутизации не задан явный маршрут. Как правило, ссылается на *маршрутизатор* локальной сети
- **Маршрутизатор** — сетевое устройство или компьютер, задача которого состоит в том, чтобы принимать трафик из одной сети и пересылать его в другую сеть
- **Машинное имя** — часть *имени узла*, уникальным образом идентифицирующая компьютер в пределах домена. См. также *Доменное имя*
- **Машинный код**. См. *Машинный язык*
- **Машинный язык** — низкоуровневые коды, понимаемые компьютером. На таком уровне разработка программ практически не ведется.

- **Многозадачность вытесняющая** – способ организации многозадачной работы при котором операционная система определяет, сколько процессорного времени получает в свое распоряжение каждая программа. Применяется во FreeBSD См также *Многозадачность кооперативная*
- **Многозадачность кооперативная** – способ организации многозадачной работы, при котором каждая программа захватывает процессор на столько времени сколько ей необходимо, после чего передает управление компьютером другим программам См также *Многозадачность вытесняющая*
- **Монтирования точка** — каталог, через который осуществляется доступ к файловой системе Обращение к файлам и каталогам файловой системы происходит так, как если бы они существовали в каталоге точки монтирования
- **Отказ от обслуживания** — тип компьютерной атаки направленной на то, чтобы лишить жертву возможности использовать свои компьютеры или сетевые соединения. При распределенной атаке злоумышленник задействует множество компьютеров (возможно, взломанных ранее) для нападения на выбранную жертву
- **Открытый исходный код** — программное обеспечение, разработаниое и распространяемое на основе философии, проповедующей доступность исходного кода программы, свободную модификацию и распространение этого кода, а также ряд других идеалов. FreeBSD — одна из ведущих операционных систем с открытым исходным кодом.
- **Офисный пакет** — набор программ, предназначениых для решения распространенных офисных задач Содержит не менее трех программ следующих категорий: текстовый процессор, редактор электронных таблиц СУБД, редактор презентаций и графический редактор.
- **Очередь на печать** — область временного хранения файлов направляемых на печать
- **Переадресация** — передача результатов работы программы в файл или другой программе, а также получение ею в качестве входных данных содержимого файла или результатов работы другой программы. Используется для организации совместной работы программ
- **Переменная среды** — системная установка доступная любой программе. Переменные среды часто используются для задания каталогов поиска исполняемых файлов и глобальных параметров проверяемых конкретными программами
- **Переносимое приложение** — программа, которая может быть перенесена из одной операционной системы в другую
- **Переносимых приложений коллекция** — совокупность файлов, содержащих информацию о получении и компиляции большого числа программ во FreeBSD Обеспечивает полуавтоматизированную инсталляцию программ
- **Подкачки область** — область диска, интерпретируемая операционной системой как расширение ОЗУ. Дает возможность компьютеру выполнять больше программ, чем позволяют физические ресурсы оперативной памяти однако это происходит медленнее, чем если бы в системе имелся эквивалентный объем дополнительной физической памяти
- **Пользовательский режим** — непривилегированный режим выполнения процесса вне ядра Пользовательские процессы управаяются ядром и только через него получают доступ к памяти, аппаратным устройствам и другим системным ресурсам См. также *Ядра режим*

- **Порт** — применительно к компьютерным сетям означает ресурс создаваемый стеком сетевых протоколов Сетевая программа подключается к порту для того, чтобы принимать и отправлять данные независимо от других программ.
- **Постраничная разбивка программа** — тип программы, отображающей текстовый файл по одному экрану за раз Во FreeBSD две наиболее распространенные программы этого типа — `more` и `less`
- **Прокси-сервер** — тип *брандмауэра*, функционирующего в виде сервера, которы частично обрабатывает поступающий запрос генерирует новый запрос к адре сату и передает возвращаемые им данные клиенту Способен кэшировать и мо дифицировать проходящие данные, что повышает скорость обработки запросов и позволяет фильтровать данные, например удалять рекламные объявления или непристойное содержимое Web-узлов
- **Протокол доставки по запросу** протокол, по которому клиент загружает со общение или документ В качестве примера можно назвать протоколы элек тронной почты POP и IMAP
- **Протокол принудительной доставки** протокол, по которому клиент отправляет сообщение или документ В качестве примера можно назвать протокол элек тронной почты SMTP
- **Процесс** — выполняющаяся программа См также *PID*
- **Рабочий стол виртуальный** — один из нескольких экранов, обслуживаемых диспетчером окон или графической оболочкой, каждый из которых содержит собственный набор окон Можно переключаться между виртуальными рабочи ми столами с помощью специальных сочетаний клавиш или команд меню На личие виртуальных экранов позволяет упорядочить работу с большим количе ством графических программ
- **Раздел** — область жесткого диска, доступная отдельно от других областей В разделах обычно хранятся файловые системы, хотя это могут быть и другие разделы, а также область подкачки.
- **Разделов таблица** — структура данных, описывающая расположение разделов на жестком диске. На платформе *x86* основная таблица разделов является ча стью *I33*, а вспомогательные таблицы хранятся в расширенных разделах или в первичных разделах FreeBSD
- **Распознавание** — преобразование имени узла например `www threeroomco com` в IP адрес и наоборот
- **Растр** — способ представления данных, при котором каждому биту данных со ответствует определенный элемент информации Растрирование широко при меняется для кодирования графических изображений, когда битам ставятся в соответствие пиксели. В случае цветных изображений каждый пиксель опреде ляется несколькими битами
- **Ретранслятор почтовый** — компьютер, принимающий почту от одной системы для доставки в другую систему. При неправильной конфигурации становится открытым ретранслятором и может быть использован для рассылки *спама* т к разрешает любому компьютеру пересылать через него почту
- **Родительский** - в контексте процессов FreeBSD означает процесс, запустив ший искомый процесс У каждого процесса есть ровно один родительский процесс См также *Дочерний*
- **Сглаживание** — техника отображения шрифта, при которой кривые и диаго нальные линии рисуются не только черными и белыми, но и серыми пикселя ми В некоторых случаях это позволяет улучшить разборчивость шрифта

- **Сервер.** (1) Программа, обрабатывающая сетевые запросы на передачу данных В качестве примеров можно назвать Web-серверы, почтовые серверы и файловые серверы (2) Компьютер, основное назначение которого заключается в том, что на нем работает одна или несколько серверных программ
- **Сетевая маска** — число, определяющее границу IP адреса которая разделяет его на адрес сети и адрес компьютера в этой сети
- **Сетевой стек** — набор протоколов, позволяющий компьютерам взаимодействовать друг с другом по сети Наиболее распространенный стек — *TCP/IP*.
- **Сигнал** — код, который можно послать процессу, чтобы вмешаться в его работу Существуют сигналы позволяющие уничтожать процессы, вышедшие из под контроля
- **Социотехника** — тип атаки, при которой злоумышленник обманным путем заставляет пользователя раскрыть конфиденциальную информацию например пароль.
- **Спам** — почтовые сообщения рекламно навязчивого характера массово рассылаемые пользователям независимо от их желания.
- **Список воспроизведения** — список аудиофайлов или записей компакт-диска которые требуется воспроизвести в ходе одного сеанса. Такие списки чаще всего ведутся проигрывателями MP3-файлов и компакт-дисков
- **Ссылка** — элемент каталога являющийся указателем на файл, который, в свою очередь связан с другим файлом Бывает двух типов: жесткая и символическая
- **Ссылка жесткая** — тип ссылки, когда два разных каталожных элемента указывают на один и тот же индексный дескриптор. В сущности их можно считать разными именами одного файла
- **Ссылка символическая** — тип ссылки, когда каталожный элемент указывает не на индексный дескриптор, а на другой каталожный элемент В данном случае адресуемый объект может находиться в другом разделе. Доступ по символической ссылке осуществляется чуть медленнее, чем в случае жесткой ссылки
- **Структура данных** — описание способа хранения данных в файле или в памяти При написании программ следует заранее указывать, с какими структурами данных им предстоит работать.
- **Суперпользователь** См *Администратор системный*
- **Суперсервер** — сервер, принимающий запросы на подключение к другим серверам и запускающий соответствующие программы Способствует повышению безопасности за счет применения единообразных проверок ко всем подконтрольным серверам, в некоторых случаях позволяет снизить потребление памяти. Во FreeBSD стандартным суперсервером является демон `inetd`
- **Сценарий** — программа, написанная на языке, который не требует компиляции Сценарии используются при конфигурировании, а также для объединения нескольких скомпилированных программ в некую функциональную оболочку
- **Сценарий интерпретатора команд** - сценарий, написанный на языке интерпретатора команд, например `sh`, `bash` или `tcsh` Такие сценарии применяются для запуска системы и решения различных административных задач.
- **Тег** — мнемонический код, содержащийся в файле или структуре данных и имеющий особое значение. Например, в *HTML*-файлах теги обозначают начало и конец гиперссылок, выделений, заголовков и т.д
- **Терминал ввода-вывода** — клавиатура и монитор, подключенные к компьютеру через последовательный порт В 70 80 гг. прошлого века такие терминалы ши

роко применялись для доступа к UNIX системам и FreeBSD до сих пор их поддерживает.

- **Терминальная программа** — программа, позволяющая компьютеру эмулировать *терминал ввода-вывода*. Часто используется для организации доступа к UNIX и FreeBSD-системам через последовательный порт или модем.
- **Топология звездообразная** — сетевая топология, когда все компьютеры соединяются в центральной точке (*концентратор* или *коммутатор*). Применяется в современных сетях Ethernet со скоростью передачи данных 100 Мбит/с.
- **Топология шинная** — сетевая топология, когда компьютеры соединяются друг с другом последовательно, образуя логическую "цепь". Применялась в старых сетях Ethernet на основе тонкого и толстого коаксиальных кабелей.
- **Троянский конь** — программа, предназначенная, по заявлениям разработчиков, для одних целей, но вместо этого или помимо этого скрыто выполняющая злонамеренные функции. Одно из средств получения несанкционированного доступа к компьютеру.
- **Утечка памяти** — ошибка, заставляющая программу запрашивать память в объемах, зачастую многократно превышающих нужную ей на самом деле. Может привести к исчерпанию ресурсов памяти компьютера.
- **Учетная запись** — набор структур данных, позволяющих пользователю получить доступ в систему.
- **Файл журнальный** — файл, в который серверы и другие системные процессы записывают информацию о своей деятельности. Большинство журнальных файлов FreeBSD находится в каталоге `/var/log`. Наиболее часто используемый журнальный файл — `/var/log/messages`.
- **Файл с точкой** — файл, имя которого начинается с точки ('.'). Такие файлы не появляются в листингах, выводимых командой `ls` и ее аналогами, если не задать специальную опцию (`-a` в случае команды `ls`). К этой категории относятся многие конфигурационные файлы пользователей.
- **Файл устройства** — особый тип файлов, позволяющих пользовательским программам получать доступ к аппаратным ресурсам компьютера. По существующему соглашению такие файлы находятся в каталоге `/dev`.
- **Файловая система** (1) Набор низкоуровневых структур данных, позволяющих операционной системе хранить данные на диске. (2) Единая каталожная структура, включающая файлы и подкаталоги.
- **Фильтр пакетов** — тип *брандмауэра*, оперирующего отдельными сетевыми пакетами. Позволяет разрешать и запрещать трафик на основании IP-адресов отправителя и получателя, номеров портов, интерфейсов и схожих низкоуровневых критериев.
- **Хакер** (1) Человек, стремящийся досконально разобраться в компьютерах, особенно в программировании, с тем чтобы применить полученные знания на практике, как правило, в полезных целях. В этом смысле многие разработчики FreeBSD являются хакерами. (2) *Взломщик*. Такая интерпретация широко распространена в средствах массовой информации, но непопулярна в сообществе разработчиков программ с открытым исходным кодом, где термин "хакер" в первоначальном значении употреблялся задолго до того, как средства массовой информации придали ему иную трактовку.
- **Хеширование** — одностороннее шифрование, не предполагающее дешифрование. В хешированном виде часто хранятся пароли, поскольку расшифровать такой пароль теоретически невозможно (на практике взломщики поступают

так: берется словарь возможных паролей, которые последовательно хешируются с применением того же алгоритма и сравниваются с искомым образцом. если обнаруживается совпадение, пароль отгадан)

- **Широковещание** — применительно к компьютерным сетям означает передачу трафика всем компьютерам сетевого сегмента. Широковещательные пакеты позволяют компьютерам находить своих соседей, а также используются в некоторых сетевых протоколах (например *DHCP*)
- **Шифрование с открытым ключом** — метод кодирования данных, подразумевающий, что один из участников обмена данными делает свой ключ шифрования доступным всем желающим. Этот ключ позволяет шифровать данные, которые могут быть расшифрованы секретным ключом получателя, а также дешифровать данные, зашифрованные таким секретным ключом
- **Шлюз**. См. *Маршрутизатор.*
- **Шрифт** — совокупность символов (чаще всего букв, цифр и знаков препинания) определенного начертания
- **Шрифт контурный**. См. *Шрифт масштабируемый*
- **Шрифт масштабируемый** — шрифт, описываемый математическим способом, при котором каждый символ определяется в виде совокупности линий и кривых. Хорошо масштабируется при любом размере и разрешении, поэтому является более гибким по сравнению с растровым аналогом
- **Шрифт растровый** — шрифт, в котором каждый символ задается в виде битового образа. Растровые символы проще обрабатывать, но они плохо масштабируются, поэтому для поддержки различных размеров шрифтов и устройств вывода с разным разрешением требуется создавать по нескольку файлов на каждое семейство шрифтов.
- **Шрифта сглаживание**. См. *Сглаживание*
- **Ядра режим** — привилегированный режим выполнения процесса как части ядра. Такие процессы имеют неограниченный доступ к системным ресурсам и другим процессам ядра, поэтому содержащиеся в них ошибки особенно опасны. См. также *Пользовательский режим*
- **Ядро** — базовый программный компонент операционной системы. Управляет аппаратными устройствами, дисковыми файловыми системами, доступом к памяти и т.д

ОТКРЫТАЯ ЛИЦЕНЗИЯ GNU

Версия 2, июнь 1991 года

Copyright © 1989, 1991 Free Software Foundation, Inc
675 Mass Ave, Cambridge MA 02139, USA
Разрешается копирование и распространение точных копий этого документа но за прещается вносить какие бы то ни было изменения в его текст

Преамбула

Для большинства программ лицензии разрабатываются с целью запрещения их свободного распространения и внесения в них изменений. Данная общая открытая лицензия GNU (GNU General Public License), наоборот, призвана гарантировать всем пользователям свободное обладание и изменение свободно распространяемого про граммного обеспечения. Эта лицензия применяется к большинству программных продуктов организации FSF и любым другим программам, авторы которых берут на себя обязательство придерживаться ее. (Существует также ряд программных продуктов FSF, которые подчиняются правилам "библиотечной" лицензии GNU — GNU Library General Public License) Вы можете применять ее к своим собственным программам.

Когда мы говорим о свободно распространяемом программном обеспечении, то имеем в виду свободу, а не цену. Наши открытые лицензии разрабатываются для того, чтобы предоставить вам свободу в распространении копий свободно распространяемых программ (при желании вы можете назначить цену за эту услугу); чтобы предос тавить вам гарантию (или возможность) получения исходных кодов; чтобы вы могли вносить изменения в программу или использовать ее фрагменты для создания новых свободно распространяемых программ, а также для того, чтобы вы знали о возможно сти реализации всех перечисленных выше действий

Для защиты ваших прав нам необходимо очертить круг ограничений которые за прещают кому бы то ни было отказать вам в этих правах иляи просить вас отказаться от своих прав. Эти ограничения налагают на вас определенные обязательства если вы распространяете копии программного обеспечения или модифицируете его

Например, при распространении копии такой программы (бесплатно или за неко торую плату) вы должны передать получателям все права, которыми обладаете сами Вы должны убедиться в том, что они также получат или могут получить исходный код Кроме того, вы должны показать им текст данной лицензии, чтобы они тоже знали свои права.

Мы защищаем ваши права двумя способами: 1) обеспечивая авторское право на программный продукт; 2) предлагая эту лицензию, которая дает вам законное разре шение на копирование, распространение и/или модификацию программы

Кроме того, в целях защиты всех авторов (и нас в том числе) мы хотим чтобы все четко понимали. свободно распространяемая программа свободна от каких бы то ни быхо гарантий. Иными словами, если программа модифицируется другими людьми и передается для последующих модификаций, то те, кому она попадает в руки должны знать, что они получили не оригинал. Поэтому любые изменения, внесенные другими пользователями, не должны отразиться на репутации авторов.

И последнее Любая свободно распространяемая программа постоянно находится под угрозой получения патента Мы хотим избежать ситуации, при которой распро странители могли бы приобрести лицензию на программу став ее владельцем Во из бежание этого мы подчеркиваем, что любой патент должен быть лицензирован для беспрепятственного использования программы всеми желающими или не лицензиро ван вообще.

Ниже приводится подробное описание условий для копирования распространения и модификации программных продуктов

УСЛОВИЯ КОПИРОВАНИЯ, РАСПРОСТРАНЕНИЯ И МОДИФИКАЦИИ ПРОГРАММНЫХ ПРОДУКТОВ

0. Данная лицензия применяется к любой программе или другому продукту, кото рый содержит замечание, внесенное владельцем авторских прав, где указано, что дан ный продукт может распространяться только на условиях общей открытой лицензии GNU. Термин "Программа" относится к любой такой программе или продукту, а вы ражение "продукт основанный на Программе" означает либо саму Программу либо любой производный продукт для которого соблюдается авторское право, т.е продукт содержащий Программу или ее часть в неизменном виде либо с модификациями и/или в переводе на другой язык (в дальнейшем возможность перевода подразумева ется в термине "модификация"). Ко всем владельцам лицензий используется обраще ние во втором лице: вы, вам, ваш и т.д. Данная лицензия не распространяется на дей ствия, отличные от копирования, распространения и модификации. На запуск Про граммы ограничения не налагаются, а результатов ее работы они касаются только в том случае, если эти результаты представляют собой продукт основанный на Про грамме. Справедливость этого положения зависит от того что делает Программа

1 Вы можете копировать или распространять точные копии полученного вами ис ходного кода Программы на любом носителе с нанесением на каждую копию соответ ствующего замечания об авторских правах и отказе от предоставления гарантии — при условии, что вы сохраните в точности все замечания в которых дается ссылка на эту лицензию и на отсутствие гарантии а также передадите другим лицам копию этой лицензии вместе с Программой

Вы можете назначить цену за услугу по передаче копии, а также на свое усмотре ние предложить гарантийную защиту в обмен на определенную сумму.

2. Вы имеете право модифицировать свою копию (копии) Программы или любую ее часть создавая таким образом продукт основанный на Программе, а также копи ровать и распространять такие модификации при соблюдении условий п. 1 соблюдая при этом следующие требования:

а) вы должны внести в модифицированные файлы замечания, которые нельзя не заметить и в которых сообщается о том, что вы изменили файлы с обяза тельным указанием даты каждого изменения,

б) согласно условиям лицензии, вы должны бесплатно лицензировать любой продукт, который вы распространяете или публикуете и который содержит Программу целиком или частично либо является продуктом, производным от Программы или от одной из ее частей

в) если модифицированная программа обычно читает команды в интерактив ном режиме, вы должны заставить ее сразу после запуска распечатать или ото бразить на экране объявление, содержащее соответствующее замечание об ка торских правах и отсутствии гарантии (или, наоборот предоставлении гаран тии), извещение о том, что пользователи могут распространять данную про грамму при соблюдении этих условий, а также указание, разъясняющее, каким образом пользователь может получить копию данной лицензии (Исключение: если Программа сама по себе работает в интерактивном режиме, но обычно не выводит подобные объявления, ваш продукт основанный на Программе не должен в обязательном порядке выводить указанное объявление)

Эти требования относятся в целом к модифицированному продукту Если в этом продукте есть разделы, которые не являются производными от Программы и могут рассматриваться как независимые продукты, работающие отдельно от других частей, то данная лицензия и ее условия не относятся к таким разделам, когда вы распро страняете их как отдельные продукты Но если вы распространяете те же самые раз

делы в качестве составных частей одного целого, которые образуют продукт основанный на Программе, распространение такого "целого" должно происходить на условиях данной лицензии, ограничения которой в этом случае расширяются на все составные части продукта, независимо от того, кто является их автором. Цель этого раздела лицензии состоит не в том чтобы оспаривать ваши права на продукт, полностью написанный вашими руками, а в заявлении права на осуществление контроля над распространением унаследованных или коллективных продуктов, основанных на Программе.

Необходимо также отметить, что простое объединение других продуктов, не основанных на Программе, с Программой (или с продуктом, основанным на Программе) на одном носителе информации или распространение такого носителя не вносит другие продукты в область действия данной лицензии.

3 Разрешается копировать и распространять Программу (или продукт, основанный на Программе, при соблюдении требований п 2) в объектном коде или исполняемом виде при соблюдении условий пп. 1 и 2 следуя одному из нижеперечисленных требований:

> а) приложите к Программе соответствующий исходный код, который должен распространяться при соблюдении условий пп. 1 и 2 на носителях, обычно ис пользуемых для обмена программными продуктами;
>
> б) приложите к Программе записанное предложение (действительное в течение по крайней мере трех лет) передать любым независимым лицам за плату не превышающую стоимость физического выполнения операции доставки, пол ную машинную копию соответствующего исходного кода, распространяемого при соблюдении условий пп 1 и 2 на носителях, обычно используемых для обмена программными продуктами
>
> в) приложите к Программе информацию, полученную вами в качестве предло жения распространять соответствующий исходный код. (Эта альтернатива разрешена только для некоммерческой поставки и только в том случае, если вы получили Программу в объектном коде или в исполняемом виде с таким пред ложением в соответствии со вторым подпунктом.)

Исходный код Программы является предпочтительной формой для внесения из менений Для исполняемого продукта полный исходный код означает наличие исход ного кода всех модулей, содержащихся в продукте, плюс соответствующие интёр фейсные файлы и сценарии, используемые для управления компиляцией и установкой исполняемого файла Но в виде исключения поставляемый исходный код не требует включения никаких компонентов, которые обычно распространяются (в исходном либо двоичном коде) вместе с основными составляющими (компилятор, ядро и т д) операционной системы, где должен работать продукт, если только какой-то компонент сам не является обязательным приложением к исполняемому продукту.

Если поставка исполняемого или объектного кода выполнена в виде предложения обратиться к копии в обозначенном месте, то предложение эквивалентного доступа к копии исходного кода из того же самого источника расценивается как поставка ис ходного кода, даже если получателя не заставляют копировать исходный код вместе с объектным кодом

4 Запрещается копировать, модифицировать, лицензировать или распространять Программу в обход этой лицензии Любые попытки поступить иначе с целью копирования, модификации, лицензирования или распространения Программы означают автоматическое аннулирование ваших прав, предусматриваемых лицензией. Но лица получившие от вас копии или права в соответствии с лицензией, не лишаются своих лицензий до тех пор, пока они полностью соблюдают оговоренные лицензией условия.

5 От вас не требуется принимать условия лицензии, пока вы не подпишете ее Но имейте в виду что ничто иное не даст вам разрешение на модификацию или распро

странение Программы или продуктов основанных на Программе Эти действия запрещены законом если вы не примете данную лицензию. Следовательно, модифицируя или распространяя Программу (или любой продукт, основанный на Программе), вы подтверждаете принятие этой лицензии и всех ее условий и требований которые необходимо соблюдать при копировании распространении и модификации Программы или продуктов, основанных на ней

6 Каждый раз, когда вы передаете Программу (или любой продукт основанный на Программе) третьему лицу, оно автоматически получает от вас лицензию на право копировать, распространять или модифицировать Программу на указанных условиях Вы не можете налагать какие бы то ни было дальнейшие ограничения на реализацию прав, данных получателю. Вы не несете ответственность за соблюдение другими лицами условий данной лицензии

7 Если вследствие решения суда или заявления о нарушении прав или по какой то другой причине (не связанной с вопросами авторских прав) поставленные перед вами условия (по постановлению суда, соглашению и т.п.) противоречат условиям данной лицензии, это не освобождает вас от ответственности за несоблюдение правил, предусматриваемых лицензией. Если вы не можете распространять Программу так, чтобы одновременно удовлетворять условиям, предусмотренным этой лицензией и соблюдать другие обязательства, значит, вы вообще не можете заниматься распространением Программы Например, если согласно некоторой лицензии не разрешается безгонорарное распространение Программы всеми, кто прямо или косвенно получает от вас ее копии, то единственный способ удовлетворить условия обеих лицензий заключается в полном отказе от распространения Программы

Если какая нибудь часть этого раздела не согласуется или не соблюдается при определенных обстоятельствах то предполагается, что оставшаяся часть раздела и весь раздел в целом применяются при других обстоятельствах

Целью этого раздела отнюдь не является склонять вас к нарушению других правовых обязательств или оспаривать их обоснованность Единственная цель этого раздела состоит в защите целостности системы свободного распространения программных продуктов, которая реализуется с помощью открытых лицензий. Многие люди внесли огромный вклад в программное обеспечение, распространяемое через эту систему, будучи уверенными в ее постоянном применении. Только от решения самого автора зависит, будет ли он (или она) распространять программный продукт через какую-либо другую систему, и лицензия не может повлиять на этот выбор.

Предполагается, что в этом разделе внесена ясность в то, что считается следствием остальной части лицензии

8 Если распространение и/или использование Программы ограничивается в определенных странах патентами либо интерфейсами, защищенными авторскими правами, исходный владелец авторских прав, который вводит свою Программу под защиту этой лицензии, может добавить в явном виде ограничение на географическое распространение, перечислив страны, исключающиеся из области распространения (это значит что распространение разрешено только в пределах тех стран, которые не входят в этот список) В этом случае лицензия включает ограничение как если бы оно было записано в теле самой лицензии

9 Организация FSF время от времени может публиковать модифицированную и/или новую версию открытой лицензии GNU. Новые версии будут воплощать ту же идею, что и настоящая версия, но могут отличаться в деталях связанных с новыми проблемами или концепциями

Каждой версии присваивается отличительный номер. Если в Программе указан номер версии лицензии, который применяется к этой и "любым последующим версиям", у вас есть возможность следовать требованиям либо данной версии либо любой из последующих версий опубликованных организацией FSF. Если в Программе не

указан номер версии лицензии вы можете выбрать любую версию, когда либо опубликованную организацией FSF.

10. Если вы хотите объединить части Программы с другими свободно распространяемыми программами, условия распространения которых отличаются от описываемых обратитесь к автору с просьбой о разрешении. Относительно программных продуктов, которые защищаются авторскими правами FSF, обращайтесь непосредственно в организацию FSF Иногда мы делаем исключения На наше решение влияет желание достичь двух целей сохранить свободный статус всех продуктов, производных от наших свободно распространяемых программ, и продвинуть идеи совместного применения и многократного использования программ

ГАРАНТИЯ ОТСУТСТВУЕТ

11. ПОСКОЛЬКУ ПРОГРАММА ЛИЦЕНЗИРУЕТСЯ БЕСПЛАТНО, ДЛЯ НЕЕ НЕ СУЩЕСТВУЕТ НИКАКИХ ГАРАНТИЙ (ДО СТЕПЕНИ, РАЗРЕШЕННОЙ ДЕЙСТВУЮЩИМ ЗАКОНОДАТЕЛЬСТВОМ). ЗА ИСКЛЮЧЕНИЕМ СЛУЧАЕВ, СПЕЦИАЛЬНО ОГОВОРЕННЫХ В ПИСЬМЕННОМ ВИДЕ, ВЛАДЕЛЬЦЫ АВТОРСКИХ ПРАВ И/ИЛИ ДРУГИЕ ЛИЦА ПРЕДОСТАВЛЯЮТ ПРОГРАММУ КАК ЕСТЬ", БЕЗ КАКОЙ ЛИБО ГАРАНТИИ, ЯВНОЙ ИЛИ ПОДРАЗУМЕВАЕМОЙ В ТОМ ЧИСЛЕ (НО НЕ ТОЛЬКО) ПОДРАЗУМЕВАЕМЫХ ГАРАНТИЙ КАЧЕСТВА И ПРИГОДНОСТИ К КОНКРЕТНОМУ ПРИМЕНЕНИЮ ВЕСЬ РИСК, СВЯЗАННЫЙ С КАЧЕСТВОМ И ВЫПОЛНЕНИЕМ ПРОГРАММЫ, ЛОЖИТСЯ НА ВАС В СЛУЧАЕ ДЕФЕКТНОСТИ ПРОГРАММЫ ВАМ СЛЕДУЕТ ВЗЯТЬ НА СЕБЯ СТОИМОСТЬ ВСЕХ НЕОБХОДИМЫХ ДОРАБОТОК ПОИСКА НЕИСПРАВНОСТЕЙ И КОРРЕКТИРОВКИ.

12. НИ ПРИ КАКИХ ОБСТОЯТЕЛЬСТВАХ, ЕСЛИ ТОГО НЕ ТРЕБУЕТ ЗАКОН ИЛИ НЕ УКАЗАНО В ПИСЬМЕННОМ ВИДЕ, ВЛАДЕЛЕЦ АВТОРСКИХ ПРАВ ЛИБО ДРУГОЕ ЛИЦО КОТОРОЕ ИМЕЛО ПРАВО МОДИФИЦИРОВАТЬ И/ИЛИ РАСПРОСТРАНЯТЬ ПРОГРАММУ В СООТВЕТСТВИИ С ПРИВЕДЕННЫМИ ВЫШЕ ТРЕБОВАНИЯМИ, НЕ НЕСЕТ ПЕРЕД ВАМИ ОТВЕТСТВЕННОСТИ ЗА ПРИЧИНЕННЫЙ ПРОГРАММОЙ УЩЕРБ ВКЛЮЧАЯ ЛЮБЫЕ ПОВРЕЖДЕНИЯ ОБЩЕГО, СПЕЦИАЛЬНОГО, СЛУЧАЙНОГО ИЛИ КОСВЕННОГО ХАРАКТЕРА, ЯВЛЯЮЩИЕСЯ СЛЕДСТВИЕМ ИСПОЛЬЗОВАНИЯ ИЛИ НЕВОЗМОЖНОСТИ ИСПОЛЬЗОВАНИЯ ПРОГРАММЫ (В ТОМ ЧИСЛЕ, НО НЕ ТОЛЬКО ПОТЕРЯ ДАННЫХ ВАМИ ИЛИ ДРУГИМИ ЛИЦАМИ, НЕПРАВИЛЬНОЕ ПРЕДСТАВЛЕНИЕ ДАННЫХ ИЛИ НЕСПОСОБНОСТЬ ПРОГРАММЫ РАБОТАТЬ СОВМЕСТНО С ДРУГИМИ ПРОГРАММАМИ), ДАЖЕ ЕСЛИ ВЛАДЕЛЕЦ ИЛИ ДРУГОЕ ЛИЦО БЫЛИ УВЕДОМЛЕНЫ О ВОЗМОЖНОСТИ ПОДОБНОГО УЩЕРБА

КОНЕЦ УСЛОВИЙ

Приложение: как применить эти требования к новым программам

Если вы написали новую программу и хотите, чтобы любой человек смог ею свободно воспользоваться, лучше всего присвоить ей статус бесплатно распространяемого программного продукта Тогда любой желающий в рамках приведенной выше лицензии сможет ее свободно распространять и модифицировать.

Для этого включите в программу приведенный ниже текст Лучше всего поместить его в качестве комментария в начало каждого исходного файла, чтобы при просмотре файла текст об отсутствии гарантий отображался в первую очередь. В каждый исходный файл должна быть как минимум включена строка "copyright", а также ссылка на то где можно прочитать полный текст лицензионного соглашения

<В первой строке необходимо указать название программы и идею ее создания >Copyright (C) год, автор

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE See the GNU General Public License for more details

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc , 675 Mass Ave Cambridge, MA 02139, USA

Кроме того, обязательно укажите свои координаты почтовый адрес и/или адрес электронной почты

Если программа работает в интерактивном режиме, сделайте так чтобы при запуске выводилось короткое сообщение наподобие приведенного ниже.

Gnomovision version 69 Copyright (C) год, автор Gnomovision comes with ABSOLUTELY NO WARRANTY, for details type show w'. This is free software, and you are welcome to redistribute it under certain conditions; type show c' for details.

С помощью упомянутых гипотетических команд show w и show c' пользователь может просмотреть соответствующие разделы открытой лицензии GNU Конечно имена команд могут быть другими Более того, эти команды могут вызываться из меню или в результате щелчка мышью — это зависит от типа программы

При необходимости укажите также координаты работодателя (если вы работаете программистом) или учебного заведения (если вы студент), которые смогут подтвердить полный отказ от всех авторских прав на программу. Ниже приведен пример, где можно вставить в текст реальные имена

Yoyodyne, Inc., hereby disclaims all copyright interest in the program Gnomovision (which makes passes at compilers) written by James Hacker.

Личная подпись дата

Должность

Данное лицензионное соглашение не предусматривает использования программы или ее частей в коммерческих проектах Если ваше программное обеспечение представляет собой библиотеку подпрограмм вы можете разрешить подключать ее компоненты к коммерческим программам При этом вместо общей открытой лицензии GNU используйте библиотечную лицензию

ЛИЦЕНЗИОННОЕ СОГЛАШЕНИЕ BSD

Вся документация и программное обеспечение, входящие в состав выпусков 4 4BSD и 4.4BSD-Lite, охраняются авторским правом организации The Regents of the University of California

Copyright 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994

The Regents of the University of California Все права защищены

Повторное распространение и использование в исходной и бинарной формах с модификацией или без, разрешается при соблюдении следующих условии

1. При повторном распространении исходного кода должны сохраняться вышеука занное замечание об авторских правах, настоящим перечень условий и нижесле дующее заявление об отсутствии ответственности
2. При повторном распространении в бинарной форме вышеуказанное замечание об авторских правах, настоящий перечень условий и нижеследующее заявление об отсутствии ответственности должны воспроизводиться в документации и/или других материалах, включаемых в дистрибутив
3. Ни название университета, ни названия вспомогательных разработчиков не могут использоваться для рекламы или продвижения продуктов, производных от данного программного обеспечения, без соответствующего предварительного письменного разрешения

ЭТО ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ ЧЛЕНАМИ ПРАВ ЛЕНИЯ УНИВЕРСИТЕТА И ВСПОМОГАТЕЛЬНЫМИ РАЗРАБОТЧИКАМИ "КАК ЕСТЬ", БЕЗ КАКОЙ-ЛИБО ГАРАНТИИ, ЯВНОЙ ИЛИ ПОДРАЗУМЕВАЕМОЙ, В ТОМ ЧИСЛЕ (НО НЕ ТОЛЬКО) ПОДРАЗУМЕВАЕМЫХ ГАРАНТИЙ КАЧЕСТВА И ПРИГОДНОСТИ К КОНКРЕТНОМУ ПРИМЕНЕНИЮ НИ ПРИ КАКИХ ОБСТОЯ-ТЕЛЬСТВАХ ЧЛЕНЫ ПРАВЛЕНИЯ ИЛИ ВСПОМОГАТЕЛЬНЫЕ РАЗРАБОТЧИКИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ЗА ПРИЧИНЯЕМЫЕ ПРОГРАММНЫМ ОБЕС ПЕЧЕНИЕМ ПРЯМЫЕ, КОСВЕННЫЕ, СЛУЧАЙНЫЕ, ОСОБЫЕ, ШТРАФНЫЕ ИЛИ ПОБОЧНЫЕ УБЫТКИ (ВКЛЮЧАЯ, НО НЕ ТОЛЬКО: ПРИОБРЕТЕНИЕ ЗАМЕЩАЮЩИХ ТОВАРОВ ИЛИ УСЛУГ, ПОТЕРЮ ПРАВА ПОЛЬЗОВАНИЯ ДАННЫХ ИЛИ ПРИБЫЛИ, НЕВОЗМОЖНОСТЬ ВЕДЕНИЯ БИЗНЕСА) КАКИ МИ БЫ ПРИЧИНАМИ ОНИ НИ ВЫЗЫВАЛИСЬ И КАКОЙ БЫ ТИП ОТВЕТ СТВЕННОСТИ БУДЬ ТО КОНТРАКТНАЯ СТРОГАЯ ИЛИ ГРАЖДАНСКАЯ (ВКЛЮЧАЯ НЕБРЕЖНОСТЬ И ПРОЧЕЕ), НИ ПРОИСТЕКАЛ ИЗ КАКОГО-ЛИБО ПРИМЕНЕНИЯ ДАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ДАЖЕ ЕСЛИ БЫЛО СДЕЛАНО УВЕДОМЛЕНИЕ О ВОЗМОЖНОСТИ ПОДОБНОГО УЩЕРБА

Предметный указатель

A

AppleTalk 365

B

BIOS 86 109 273

C

CUPS, 155
CVS, 266; 598

D

DHCP 35
  сервер 430
    инсталляция, 431
    конфигурирование 432 433
  статические адреса, 433
DNS, 308; 434
  запуск сервера 435
  конфигурирование, 312 436
  корневая зона, 437
  обратное преобразование, 439
  поиск почтовых серверов, 378; 381
  прямое преобразование, 438

E

Ethernet 51 303

F

FHS, 188, 189; 191
FTP, 364; 479; 482
  команды, 481
  сервер, 366
    запуск 367

G

GNOME, 79; 99 455
  запуск, 459
  конфигурирование, 460 464 465
  экранное меню, 465

H

HTML 399 402 404

I

IP адрес, 304; 307
  класс, 307
  маска подсети, 307
  статический, 308

K

KDE 79, 453
Kerberos, 245

L

Linux
  запуск программ во FreeBSD, 127; 128
  монтирование разделов FreeBSD, 126
  отличия от FreeBSD, 39
  режим бинарной совместимости 75

M

MAC-адрес, 307; 314
MIME 461· 541

N

NAT, 337; 348; 408
  конфигурирование 358
NFS, 364
  безопасность, 369
  сервер, 367
    запуск, 370
  экспорт каталога 484

P

PAM 164 387

S

Samba, 365; 370
  глобальные параметры 370
  запуск, 374
  каталоги совместного доступа 372
  команда smbpasswd, 372
  пароли, 372
  точки совместного доступа 486
SFTP, 479
SMB/CIFS 486

SSH, 163, 365; 406, 410
запуск сервера, 411
клиент, 489
конфигурирование, 409; 411
туннелирование X-трафика 415

T

TCP/IP, 35, 42 304
Telnet, 406
запуск сервера, 408
клиент, 489
конфигурирование, 407
ограничение доступа, 408

V

VNC, 407, 421
запуск, 426
инсталляция, 422
конфигурирование, 423, 424
установление соединения, 425

W

Web сервер 35 365 393 394

X

X Window System, 33, 98 282 407
безопасность, 571
конфигурирование, 287; 294 296
параметры клавиатуры, 297
параметры мыши, 296
работа в сети, 286
туннелирование, 415
удаленный доступ, 413
в графическом режиме, 417
в текстовом режиме, 414
шрифты, 444
XDMCP, 417
клиент, 419
сервер, 417
XFce, 456
XFree86, 50; 283, 294; 421
конфигурирование, 76
шрифты, 445
X сервер, 282, 413
конфигурирование, 76, 294, 296
параметры клавиатуры, 297
параметры мыши, 296
останов, 104
шрифты, 292
X среда, 33, 98; 282 407 413 451
запуск нескольких сеансов, 298
конфигурирование, 102, 287
набор элементов управления 283
регистрация
в графическом режиме, 417
в текстовом режиме, 414
управление файлами 100
шрифты, 444
X-терминал, 417, 420

Б

Безопасность, 238; 252; 344; 346
анализатор пакетов, 572
дистанционные атаки, 570
зондирование системы 569
локальные атаки, 571
непрямые атаки, 572
несанкционированный код 346
обнаружение взлома, 586
отказ от обслуживания, 573
переполнение буфера, 570
социотехника, 571
способы атак, 569
устранение взлома, 591
Брандмауэр, 337; 344; 345; 582
правила фильтрации, 353; 354
реализация, 351; 357
средства настройки, 349
стандартные правила, 352
типы, 347
Броузер, 35; 475
безопасность, 478
конфигурирование, 476
подключение мультимедийного проигрывателя, 541

В

Взломщик, 569
Выход из системы 104

Г

Главная загрузочная запись, 68; 109
Графическая оболочка, 285; 451; 453; 457
выбор, 296
Группа, 81, 238; 251, 253
wheel, 84; 238
идентификатор, 193 239
приватная, 252
создание, 82, 251

Д

Демон, 89
  cron, 169, 170, 171
  httpd, 395
  inetd, 137, 580
    конфигурирование 339
  init, 88, 135; 158
  lpd, 219; 224
    конфигурирование 234
  named, 435
  natd, 358
  nfsd, 370
  ntpd, 441; 443
  pppd, 325, 328
  sshd, 411
Диспетчер окон 283 451
  Afterstep, 79
  Enlightment, 79
  fvwm, 79
  GDM, 296
  IceWM, 168
  Sawfish, 79
  Windowmaker, 79
  XDM, 90
    вызов, 287
  выбор, 296
  отличительные особенности 99
Диспетчер файлов, 100; 451
  Nautilus, 461
Диспетчер экрана
  GDM, 418
  KDM, 417
  WDM, 417
  XDM, 417
Документация, 627
Доменное имя, 307

Ж

Жесткий диск
  IDE, 47
  SCSI, 47
  разбивка на разделы, 66
  устранение неполадок 636

З

Загрузка системы, 86; 109
  устранение неполадок, 640
Загрузчик системы, 49, 68, 109 110 158
  сторонний, 113
  этапа 0, 86
  этапа 1, 86; 112
  этапа 2, 86 112
Звук, 529
  MP3-файлы, 535; 536, 537
  включение поддержки, 530
  запись и воспроизведение 533 534
  микшеры, 532
  музыкальные компакт диски 537

И

Игры, 543; 544, 545
Индексный дескриптор, 147; 553
Интерпретатор команд 81· 92· 607
  язык, 608

К

Канал, 154, 611
Каталог
  /dev, 117
  /etc, 135; 136; 138
  права доступа, 197
  создание, 98
  список содержимого 95
  текущий, 95
  удаление, 98 205
Квоты, 553
Коллекция переносимых приложений 57
Команда
  adduser, 243
  alias, 167
  arp, 434
  at, 169
  batch, 169
  bc, 615
  bg, 90
  boot0cfg, 110
  brandelf, 128
  cat, 202
  cd, 95
  cd2mp3, 536
  chat, 327
  chgrp, 199
  chmod, 199
  chown, 198
  chpass, 246
  cp, 97; 201
  cpio, 211
  crontab, 172
  dd, 203
  df, 147
  disklabel 179

dmesg, 87, 146
du, 552
dump, 211
echo, 613
egrep, 208
export, 166; 612
fdisk, 116; 178
fg, 90
find, 207
fstat, 575
ftp, 482
grep, 208
gs, 227
hostname, 312
ifconfig, 310
ipfw, 350
  инструкции, 355
  примеры правил 356
  синтаксис, 354
  спецификация адреса 355
kill, 151, 556
killall, 151
kldload, 275
kldstat, 275
kldunload, 276
ldd, 128
less, 139
ln, 206
logout, 104
lpc, 233
lpq, 231
lpr, 218; 230
lprm, 232
lptcontrol 222
ls, 95
make world, 565
man, 148; 628
md5, 271
mixer, 532
mkdir, 98
more, 139
mount, 123; 126, 183 485
mount_nfs, 485
mount_smbfs, 487
mv, 97, 202
netstat, 311 341; 342 575
newfs, 181
newfs_msdos, 182
newsyslog, 550
nice, 150
ntpdate, 442
ntpq, 443
passwd, 245
ping, 315; 316
pkg_add, 262
pkg_delete, 268
pkg_info, 263, 552
pkg_update, 269
pkgdb, 270; 564
play, 533
portupgrade, 270 563
ppp, 329, 330
ps, 149
pwd_mkdb 248
rcp, 365
read, 614
rec, 533
renice, 150 557
rlogin, 163
rm, 97; 204
rmdir, 98, 205
rmuser, 249
route, 312
setenv, 167
shutdown, 105
smbclient, 486
ssh, 489
startx, 99
su, 91
sysctl, 558
tar, 211
telnet, 334, 407; 489
top, 150; 555; 559
traceroute, 316
ttmkfdir, 293
typelinst, 293
umask, 200, 253
umount, 186
uptime, 148; 554
whois, 570
xauth, 416
xset, 297; 298; 458
xsetroot, 458
Коммутатор, 304
Компилятор, 597
Консоль, 88
  виртуальная, 90
Концентратор, 304

Л

Лицензия
  BSD, 38; 56
  GPL, 39, 56

М

Маршрутизатор 304, 306
  NAT, 348
Маршрутизация, 306 311 336
  тестирование, 316
Микропроцессорный набор 60
Микроядро, 38
Модем, 52
  конфигурирование, 322
  набор номера, 320; 325
    сценарий, 327
  тестирование, 319
  управление соединением, 328
  установление соединения, 324
  файл аутентификации, 321
Мультимедиа, 538; 540

О

Область подкачки, 558, 560
Оперативная память, 46
  контроль использования, 558; 559
  устранение неполадок, 636
Останов системы, 105, 106
Открытый исходный код, 38 57
Отладчик, 598
Офисный пакет
  GNOME Office 492
  KOffice, 493
  OpenOffice org 494; 499
    векторная графика, 506
    презентации, 507
    редактирование текста, 502
    создание документа, 502
    электронные таблицы, 504
  импот/экспорт файлов, 498
  обработка шрифтов, 496
  печать, 497

П

Пароль, 81, 578
  безопасность, 583
  скрытый, 245
  создание, 245, 584
Переадресация ввода-вывода 153 611
Переменная среды, 166
  PATH, 94
Переносимое приложение 257 595
  инсталляция, 267
  коллекция, 258; 264· 265
  обновление, 270
Печать, 217
  выдача заданий, 229
  демон построчной печати 219
  из приложений, 230
  каталог спулинга, 222
  контроль заданий, 231
  область спулинга, 217
  очередь, 217, 218; 220
    изменение статуса, 233
  система BSD LPD, 218
  система CUPS, 218
  система LPRng, 218
  удаление заданий, 232
  упорядочение заданий, 233
  файлы устройств 220
  фильтр, 218, 225
    magicfilter, 228
  язык PostScript, 219, 523, 525
Пользователь
  идентификатор, 81, 193· 239
    изменение, 248
  имя, 81; 241
  начальный каталог, 81, 237 240
  учетная запись, 80, 237
Порт, 305; 333
  TCP, 334
  UDP, 334
  непривилегированный, 353
  номер, 333, 334
  привилегированный, 333; 353
Приглашение командной строки 90
Принтер, 52; 220
  конфигурирование, 223
  оптимизация порта, 222
  сетевой, 234
  тестирование, 221
  устранение неполадок 639
Программа
  Apache, 393; 394
    запуск, 395
    каталоги Web-страниц, 397
    конфигурирование, 394; 395 396
    модули, 398
  BIND, 435, 436
  Control Center, 102 460
  Dia, 519
  DNDR, 350
  Exim, 379
  FDISK Partition Editor 66
  Fetchmail, 388
    запуск, 390
    режим демона 391
  FIPS, 63

FreeBSD Disklabel Editor 68
getty, 162
gFTP, 484
Ghostscript, 53; 220; 226; 227; 524 525
GIMP, 511; 512 513; 514; 516
Gnuplot, 520
IP Filter, 350
jftpgw, 350
login, 162
OpenSSH, 410
PartitionMagic, 64
Postfix, 379
Privoxy, 350
procmail, 385
qmail, 379
SANE, 54
sendmail, 379, 380
  конфигурационные файлы 381
  конфигурирование, 381
  ретрансляция почты, 382
  спам, 383; 386
  фильтрация заголовков, 385
  фильтрация содержимого, 385
Squid, 350
SWAT, 155
sysinstall, 65; 70; 154
  загрузка кода ядра, 276
  задание часового пояса, 75
  инсталляция переносимых приложений, 265
  инсталляция программ, 260
  конфигурирование X-сервера, 76
  конфигурирование консоли, 74
  конфигурирование рабочего стола 79
  настройка параметров мыши, 75
  настройка параметров сети, 72
  настройка сетевых параметров, 309
  настройка учетных записей, 80
  создание групп, 251
  создание разделов, 176
  создание учетных записей 241
TCP Wrappers, 409, 580
Tripwire, 587
  база данных, 589
  инсталляция, 588
  конфигурирование, 588
UW IMAP, 380, 387
Webmin, 155
WINE, 129
  запуск приложений Windows 131
  конфигурирование, 129
xf86cfg, 294
XFig, 518
X-ISP, 322
запуск, 94
  в графической среде, 102
инсталляция, 260; 262, 602; 603
компиляция, 595; 596; 597; 599; 601
обновление, 269; 562
постраничной разбивки, 139
проверка аутентичности 270
путь поиска 94
разновидности 256
способы распространения 257
удаление, 268; 574; 577
Прокси-сервер 337 349 350
Протокол
  DHCP, 314
  ICMP, 335
  IMAP, 377; 380 387
  NTP, 440
    конфигурирование 442
    поиск сервера, 441
  POP, 377, 380, 387
  PPP, 325
  PPPoE, 328
    конфигурирование, 329
    управление соединением, 330
  SMTP, 377; 379; 380
  TCP, 334
  UDP, 334
  XDMCP, 162
Процесс, 135
  зависший, 556
  приоритет, 150; 557
  уничтожение, 151
Процессор, 45
  контроль загруженности, 554 556
  средняя загруженность, 148
  устранение неполадок, 634

P

Рабочая станция 33
Рабочий стол
  виртуальный, 284
  конфигурирование, 79
Раздел, 43, 114; 175; 186; 187
  демонтирование, 186
  защита от повреждения 115
  код типа, 115
  логический, 44; 115
  монтирование, 182; 187
  первичный, 44; 114
    создание, 67

расширенный, 44; 114
создание, 176; 178; 179
Регистрация в системе, 162; 166
графический режим 98; 168
дистанционная
в графическом режиме, 162
в текстовом режиме 163
консольная
в графическом режиме, 162
в текстовом режиме, 162
текстовый режим, 89; 167
Регулярное выражение, 208
Резервное копирование, 210; 211
восстановление данных, 214

С

Сервер, 305; 332
дистанционной регистрации 35 406
запуск, 338
ограничение доступа 578
почтовый, 35, 377
IMAP, 380; 387
POP, 380; 387
SMTP, 379, 380
синхронизации времени 440
конфигурирование, 442
поиск, 441
файловый, 34, 363; 364
шрифтов, 444; 446
запуск, 446
конфигурирование, 445
Сетевое оборудование, 303
устранение неполадок, 639
Сетевое соединение, 335; 336
диагностика, 341
Сетевой интерфейс, 303
localhost, 312
активизация, 310
Сетевой стек, 304
AppleTalk, 305
IPX, 306
NetBEUI, 305
Сигнал, 152
Спам, 383
входящий, 383
фильтрация заголовков, 385
фильтрация содержимого, 385
исходящий, 386
черные списки, 383
Среда разработки, 598
Ссылка
жесткая 205
символическая, 96; 205
Суперпользователь, 82, 91 196
пароль, 585
Суперсервер, 339
Сценарий, 606; 607
аргументы командной строки, 613
выполнение внешних комаид, 610
запуск, 609
код завершения, 610
переменные, 612; 613
прием данных от пользователя, 614
условные выражения, 616; 617, 619
функции, 623
циклы, 620
Сценарий запуска 88 135 160 338
локальный, 161
системный, 160

Т

Текстовый процессор, 138
Текстовый редактор, 138
Easy Editor, 139
Emacs, 139; 140
gEdit, 140, 143, 145
NEdit, 140
vi, 139
редактирование файла, 141
режимы работы, 140
сохранение изменений 143
xedit, 140
Терминал
ввода-выводы, 164
графический, 286

У

Учетная запись, 80 193 237 240
root, 82, 91; 196
пароль, 585
безопасность, 238, 252
модификация, 244; 246
создание, 83; 240; 241; 243
удаление, 249

Ф

Файл
/etc/crontab, 171
/etc/exports, 368
/etc/fstab, 184
/etc/group, 252
/etc/hosts, 313
/etc/hosts.allow 580

/etc/master.passwd, 240; 247 250
/etc/passwd, 240; 245
/etc/ppp/options, 325
/etc/ppp/ppp.conf, 329
/etc/printcap, 223
/etc/rc, 136
/etc/rc.conf, 313
/etc/resolv.conf, 312
/etc/services, 334
/etc/ttys, 162; 164; 287
/etc/X11/XF86Config, 287
конфигурирование графической платы, 290
конфигурирование дисплея, 289
конфигурирование клавиатуры, 297
конфигурирование монитора, 291
конфигурирование мыши, 292
конфигурирование шрифтов, 292
владелец, 193
изменение 198
временный, 551
журнальный, 549
индексный дескриптор 205 553
копирование, 97; 201
перемещение, 97; 201
поиск, 206
права доступа, 193; 194
sticky-бит, 197
бит SGID, 196
бит SUID, 196
изменение, 199
стандартные, 253
тип, 195
удаление, 97; 204
Файловая система, 118
ext2fs, 121
FAT, 45; 120
FFS, 45; 118; 125
разновидности 119
NTFS, 121
proc, 185
для компакт-дисков 122
журнальная, 121
монтирование, 123
ограничения, 124
создание, 181
точка монтирования 183
экспортируемая, 367
Фильтр пакетов, 347; 349; 353 354
реализация, 351; 357
стандартные правила 352
Фоновый режим, 90

X

Хакер 509

Э

Электронна почта 377
вложения, 474
клиент, 469 470
конфигурирование 470
отправка, 473
ретранслятор, 377
сервер
IMAP, 380; 387
POP, 380; 38
SMTP, 379; 380
чтение, 472
Элемент управлени 100
Эмулятор, 132

Я

Ядро, 38; 158; 273
буфер сообщений 146
загрузка, 158; 275
задание параметров, 112
звуковые модули, 529; 530
инсталляция, 274
исходный код, 276; 280
компиляция, 276; 278
конфигурирование, 277
модули, 275
обновление, 279
поддержка NAT, 358
подключение, 279
сообщения, 87; 279
фильтрация пакетов, 351 352
Языки сценариев, 608

Научно популярное издание

Родерик Смит

Полный справочник по FreeBSD

Литературный редактор	*Ж.Е. Прусакова*
Верстка	*А.Н. Полинчик*
Художественный редактор	*С.А. Чернокозинский*
Корректоры	*З.В. Александрова, Л.А. Гордиенко, О.В. Мишутина, Л.В. Чернокозинская*

Издательский дом "Вильямс".
101509, Москва, ул. Лесная, д. 43, стр. 1
Изд. лиц. ЛР № 090230 от 23.06.99
Госкомитета РФ по печати.

Подписано в печать 20.08.2004. Формат 70×100/16
Гарнитура Times. Печать офсетная.
Усл. печ. л. 42,0. Уч.-изд. л. 46,5.
Доп. тираж 3000 экз. Заказ № 277

Отпечатано с фотоформ в ФГУП "Печатный двор
Министерства РФ по делам печати
телерадиовещания и средств массовых коммуникаций
197110, Санкт-Петербург, Чкаловский пр., 15

Полный справочник по FreeBSD

Компакт-диск содержит FreeBSD 5 2 1

В чем преимущества FreeBSD? Будучи членом семейства UNIX, FreeBSD является высоконадежной, гибкой операционной системой, которую можно применять в самых разных целях. Книга может служить подробным руководством по инсталляции, конфигурированию и обслуживанию системы Читатели узнают, как настроить эту мощную операционную систему для работы с существующим аппаратным и программным обеспечением, как управлять ею и как использовать имеющиеся в ней графические и офисные приложения Являетесь ли вы пользователем-любителем или профессионалом в области информационных технологий, данная книга позволит вам изучить функционирование этой популярной операционной системы с открытым исходным кодом.

Издательский дом "Вильямс"
www.williamspublishing.com

OSSORNE
www.osborne.com

A Division of The McGraw-Hill Companies

ISBN 5 8459 0576 1